

Recurrences II

1 Asymptotic Notation and Induction

We've seen that asymptotic notation is quite useful, particularly in connection with recurrences. And induction is our favorite proof technique. But mixing the two is risky business; there is great potential for subtle errors and false conclusions!

False Claim 1. *If*

$$\begin{aligned}T(1) &= 1 \\T(n) &= 2T(n/2) + n\end{aligned}$$

then $T(n) = O(n)$.

This claim is false; the Akra-Bazzi theorem implies that the correct solution is $T(n) = \Theta(n \log n)$. But where does the following "proof" go astray?

Proof. The proof is by strong induction. Let $P(n)$ be the proposition that $T(n) = O(n)$.

Base case: $P(1)$ is true because $T(1) = 1 = O(1)$.

Inductive step: For $n \geq 2$ assume $P(1), P(2), \dots, P(n-1)$ to prove $P(n)$. We have:

$$\begin{aligned}T(n) &= 2 \cdot T(n/2) + n \\&= 2 \cdot O(n/2) + n \\&= O(n)\end{aligned}$$

The first equation is the recurrence, the second uses the assumption $P(n/2)$, and the third is a simplification. □

Where's the bug? The proof is already far off the mark in the second sentence, which defines the induction hypothesis. The statement " $T(n) = O(n)$ " is either true or false; its validity does not depend on a particular value of n . Thus, the very idea of trying to prove that the statement holds for $n = 0, 1, 2, \dots$ is wrong-headed.

The safe way to reason inductively about asymptotic phenomena is to *work directly with the definition of the notation*. Let's try to prove the claim above in this way. Remember that $f(n) = O(n)$ means that there exist constants n_0 and $c > 0$ such that $|f(n)| \leq cn$ for all $n \geq n_0$. (Let's not worry about the absolute value for now.) If all goes well, the proof

attempt should fail in some blatantly obvious way, instead of in a subtle, hard-to-detect way like the earlier argument. Since our perverse goal is to demonstrate that the proof won't work for *any* constants n_0 and c , we'll leave these as variables and assume only that they're chosen so that the base case holds; that is, $T(n_0) \leq cn$.

Proof Attempt. We use strong induction. Let $P(n)$ be the proposition that $T(n) \leq cn$.

Base case: $P(n_0)$ is true, because $T(n_0) \leq cn$.

Inductive step: For $n > n_0$, assume that $P(n_0), \dots, P(n-1)$ are true in order to prove $P(n)$. We reason as follows:

$$\begin{aligned} T(n) &= 2T(n/2) + n \\ &\leq 2c(n/2) + n \\ &= cn + n \\ &= (c+1)n \\ &\not\leq cn \end{aligned}$$

The first equation is the recurrence. Then we use induction and simplify until the argument collapses!

2 Linear Recurrences

Last lecture we saw how to solve Divide and Conquer recurrences. In this lecture, we'll consider another family of recurrences, called "linear recurrences", that also frequently arise in computer science and other disciplines. We'll first see how to solve a specific linear recurrence and then generalize our method to work for all linear recurrences.

2.1 Graduate Student Job Prospects

In a new academic field (say computer science), there are only so many faculty positions available in all the universities of the world. Initially, there were not enough qualified candidates, and many positions were unfilled. But, over time, new graduates are filling the positions, making job prospects for later computer science students ever more bleak. Worse, the increasing number of professors are able to train an increasing number of graduate students, causing positions to fill ever more rapidly. Eventually, the universities will be saturated; new computer science graduates will have no chance at an academic career. Our problem is to determine when the universities will stop hiring new computer science faculty and, in particular, to answer the question, "Are the 6.042 TAs doomed?" Here are the details of the problem.

- There are a total of N faculty positions available worldwide. This number never changes due to budgetary constraints.

- Congress has passed a law forbidding forced retirement in universities, and no one will retire voluntarily. (This is true and a problem!) In our analysis, therefore, once a faculty position is filled, it never opens up again.
- Each year, every professor trains exactly 1 student who will go on to become a professor the following year. The only exception is that first year professors do not train students; they are too busy publishing, teaching, getting grants, and serving on committees.
- In year 0, there are no computer science professors in the world. In year 1, the first professor is hired.

2.2 Finding a Recurrence

Ideally, we could find a formula for the number of professors in the world in a given year. Then we could determine the year in which all N faculty positions are filled. Let $f(n)$ be the number of professors during year n . To develop some intuition about the problem, we can compute values of this function for small n by hand.

| | |
|------------|---|
| $f(0) = 0$ | No CS professors; the dark ages. |
| $f(1) = 1$ | 1 new professor; too busy to train a student. |
| $f(2) = 1$ | 1 old professor; now training a student. |
| $f(3) = 2$ | 1 new prof, 1 old prof; new prof too busy, old prof training. |
| $f(4) = 3$ | 1 new prof, 2 old profs; new prof too busy, both old profs training |
| $f(5) = 5$ | 2 new profs, 3 old profs |
| $f(6) = 8$ | 3 new profs, 5 old profs |

In general, the number of professors in year n is equal to the number of professors last year plus the number of new hires. The number of professors last year is $f(n - 1)$. The number of new hires is equal to the number of professors two years ago, $f(n - 2)$, since each of these professors trained a student last year. These observations give the following recurrence equation for the number of professors:

$$\begin{aligned}
 f(0) &= 0 \\
 f(1) &= 1 \\
 f(n) &= f(n - 1) + f(n - 2) \quad (n \geq 2)
 \end{aligned}$$

This is the familiar Fibonacci recurrence. Looking back, the values of $f(n)$ that we computed by hand are indeed the first few Fibonacci numbers. Fibonacci numbers arise in all sorts of applications. Fibonacci himself introduced the numbers in 1202 to study

rabbit reproduction. Fibonacci numbers also appear, oddly enough, in the spiral patterns on the faces of sunflowers. And the input numbers that make Euclid's GCD algorithm require the greatest number of steps are consecutive Fibonacci numbers. So how big is $f(n)$ anyway? Of course, we could compute as many Fibonacci numbers as we like using the recurrence, but it would be much nicer to find a closed form.

2.3 Solving the Recurrence

Solving the Fibonacci recurrence is easy because the recurrence is linear. (Well, "easy" in the sense that you can learn the technique in one lecture; discovering it actually took six centuries.) A *linear recurrence* has the form:

$$\begin{aligned} f(n) &= a_1 f(n-1) + a_2 f(n-2) + \dots + a_d f(n-d) \\ &= \sum_{i=1}^d a_i f(n-i) \end{aligned}$$

where a_1, a_2, \dots, a_d are constants. The *order* of the recurrence is d . For example, the Fibonacci recurrence is order 2 and has coefficients $a_1 = a_2 = 1$. (Later on, we'll slightly expand the definition of a linear recurrence.)

For now, let's try to solve just the Fibonacci recurrence; we'll see how to solve general linear recurrences later in the lecture. Our rule of thumb is that guess-and-verify is the first method to apply to an unfamiliar recurrence. It turns out that for a linear recurrence, an exponential solution is a good guess. However, since we know nothing beyond this, our initial guess-and-verify attempt will really only be a "dry run"; that is, we will not make an exact guess and will not verify it with a complete proof. Rather, the goal of this first attempt is only to clarify the form of the solution.

Guess. $f(n) = cx^n$

Here c and x are parameters introduced to improve our odds of having a correct guess; in the verification step, we can pick values that make the proof work. To further improve our odds, let's neglect the boundary conditions, $f(0) = 0$ and $f(1) = 1$.

Verification. Plugging our guess into the recurrence $f(n) = f(n-1) + f(n-2)$ gives:

$$\begin{aligned} cx^n &= cx^{n-1} + cx^{n-2} \\ x^2 &= x + 1 \\ x^2 - x - 1 &= 0 \\ x &= \frac{1 \pm \sqrt{5}}{2} \end{aligned}$$

In the first step, we divide both sides of the equation by cx^{n-2} . Then we rearrange terms and find x with the quadratic formula.

This calculation suggests that the constant c can be anything, but that x must be $(1 \pm \sqrt{5})/2$. Evidently, there are two solutions to the recurrence:

$$f(n) = c \left(\frac{1 + \sqrt{5}}{2} \right)^n \quad \text{or} \quad f(n) = c \left(\frac{1 - \sqrt{5}}{2} \right)^n$$

□

In fact, any linear combination of these two solutions is also a solution. The following theorem states that this is true in general for linear recurrences.

Theorem 2. *If $f(n)$ and $g(n)$ are solutions to a linear recurrence (without boundary conditions), then $cf(n) + dg(n)$ is also a solution.*

Proof. Since $f(n)$ and $g(n)$ are both solutions, then:

$$f(n) = \sum_{i=1}^d a_i f(n-i)$$

$$g(n) = \sum_{i=1}^d a_i g(n-i)$$

Multiplying the first equation by c , the second by d , and summing gives:

$$cf(n) + dg(n) = c \cdot \left(\sum_{i=1}^d a_i f(n-i) \right) + d \cdot \left(\sum_{i=1}^d a_i g(n-i) \right)$$

$$= \sum_{i=1}^d a_i (cf(n-i) + dg(n-i))$$

Thus, $cf(n) + dg(n)$ is a solution as well. □

This same phenomenon— that a linear combination of solutions is another solution— also arises in differential equations and, consequently, many physical systems. In the present case, the theorem implies that

$$f(n) = c_1 \left(\frac{1 + \sqrt{5}}{2} \right)^n + c_2 \left(\frac{1 - \sqrt{5}}{2} \right)^n$$

is a solution to the Fibonacci recurrence without boundary conditions for all constants c_1 and c_2 . All that remains is to choose these constants to obtain a solution consistent with the boundary conditions, $f(0) = 0$ and $f(1) = 1$. From the first condition, we know:

$$f(0) = c_1 \left(\frac{1 + \sqrt{5}}{2} \right)^0 + c_2 \left(\frac{1 - \sqrt{5}}{2} \right)^0 = c_1 + c_2 = 0$$

From the second boundary condition, we have:

$$f(1) = c_1 \left(\frac{1 + \sqrt{5}}{2} \right)^1 + c_2 \left(\frac{1 - \sqrt{5}}{2} \right)^1 = 1$$

We now have two linear equations in two unknowns. The system of equations is not degenerate, so there is a unique solution: $c_1 = 1/\sqrt{5}$ and $c_2 = -1/\sqrt{5}$. We're done! We have a complete solution to the Fibonacci recurrence *with* boundary conditions:

$$f(n) = \frac{1}{\sqrt{5}} \left(\frac{1 + \sqrt{5}}{2} \right)^n - \frac{1}{\sqrt{5}} \left(\frac{1 - \sqrt{5}}{2} \right)^n$$

This *looks* completely wrong! All Fibonacci numbers are integers, but this expression is full of square roots of five! Amazingly, however, the square roots always cancel out. This expression really does give the Fibonacci numbers if we plug in $n = 0, 1, 2, \dots$. It is easy to see why no one stumbled across this solution for six centuries!

2.4 Job Prospects

Let's return to the original question: how long until all N faculty positions are taken?

To answer this question, we must find the smallest n such that $f(n) \geq N$; that is, we must determine the year n in which there are as many potential professors as university positions. Graduates after year n will have to find other employment, e.g. shuffling golden disks in an obscure monastic community for the next 10^{19} years.

Because $f(n)$ has such a complicated form, it is hard to compute the right value of n exactly. However, we can find an excellent approximate answer. Note that in the closed form for Fibonacci numbers, the second term rapidly goes to zero:

$$\begin{aligned} f(n) &= \frac{1}{\sqrt{5}} \left(\frac{1 + \sqrt{5}}{2} \right)^n - \frac{1}{\sqrt{5}} \left(\frac{1 - \sqrt{5}}{2} \right)^n \\ &= \frac{1}{\sqrt{5}} \left(\frac{1 + \sqrt{5}}{2} \right)^n + o(1) \end{aligned}$$

This is because $|(1 - \sqrt{5})/2| = 0.618\dots < 1$, and a big power of a number less than 1 is tiny.

From this approximation for $f(n)$, we can estimate the year in which all faculty positions will be filled. That happens when:

$$f(n) \approx \frac{1}{\sqrt{5}} \left(\frac{1 + \sqrt{5}}{2} \right)^n \geq N$$

Thus, all jobs are filled in about n years where:

$$\begin{aligned} n &= \frac{\log(\sqrt{5} N + o(1))}{\log\left(\frac{1+\sqrt{5}}{2}\right)} \\ &= \Theta(\log N) \end{aligned}$$

This makes sense, since the number of professors is increasing exponentially. For example, $N = 10,000$ jobs would all be taken in about $n = 20.8$ years. Your TAs don't have a moment to lose!

The solution to the Fibonacci recurrence has an interesting corollary. The number:

$$\gamma = \left(\frac{1 + \sqrt{5}}{2} \right)$$

is often called the “Golden Ratio”. We can write the dominant term in the closed form for the n -th Fibonacci number in terms of the γ :

$$f(n) = \frac{\gamma^n}{\sqrt{5}} + o(1)$$

We've just shown that this expression involving irrational numbers is actually very close to an integer for all large n —namely, the n -th Fibonacci number. This is just one of many curious properties of the Golden Ratio.

3 General Linear Recurrences

The method we used to solve the Fibonacci recurrence can actually be used to solve any linear recurrence. Recall that a recurrence is linear if it has the form:

$$f(n) = a_1 f(n-1) + a_2 f(n-2) + \dots + a_d f(n-d)$$

Substituting the guess $f(n) = x^n$, as with the Fibonacci recurrence, gives:

$$\begin{aligned} x^n &= a_1 x^{n-1} + a_2 x^{n-2} + \dots + a_d x^{n-d} \\ x^d &= a_1 x^{d-1} + a_2 x^{d-2} + \dots + a_{d-1} x + a_d \end{aligned}$$

Dividing the first equation by x^{n-d} gives the second. This second equation is called the *characteristic equation* of the recurrence. The characteristic equation can be read off very quickly since the coefficients of the equation are the same as the coefficients of the recurrence.

The solutions to a linear recurrence are defined by the roots of the characteristic equation. Neglecting boundary conditions for the moment:

- If r is a nonrepeated root of the characteristic equation, then r^n is a solution to the recurrence.
- If r is a repeated root with multiplicity k , then

$$r^n, \quad nr^n, \quad n^2r^n, \quad \dots, \quad n^{k-1}r^n$$

are all solutions to the recurrence.

Futhermore, Theorem 2 implies that every linear combination of these solutions is also a solution.

For example, suppose that the characteristic equation of a recurrence has roots r_1 , r_2 , and r_3 twice. These four roots imply four distinct solutions:

$$f(n) = r_1^n \quad f(n) = r_2^n \quad f(n) = r_3^n \quad f(n) = nr_3^n$$

Thus, every linear combination

$$f(n) = a \cdot r_1^n + b \cdot r_2^n + c \cdot r_3^n + d \cdot nr_3^n$$

is also a solution.

All that remains is to find a solution consistent with the boundary conditions by choosing the constants appropriately. Each boundary condition implies a linear equation involving these constants. So we can determine the constants by solving a system of linear equations. For example, suppose our boundary conditions were $f(0) = 0$, $f(1) = 1$, $f(2) = 4$ and $f(3) = 9$. Then we would obtain four equations in four unknowns:

$$\begin{aligned} f(0) = 0 &\quad \Rightarrow \quad a \cdot r_1^0 + b \cdot r_2^0 + c \cdot r_3^0 + d \cdot 0r_3^0 = 0 \\ f(1) = 1 &\quad \Rightarrow \quad a \cdot r_1^1 + b \cdot r_2^1 + c \cdot r_3^1 + d \cdot 1r_3^1 = 1 \\ f(2) = 4 &\quad \Rightarrow \quad a \cdot r_1^2 + b \cdot r_2^2 + c \cdot r_3^2 + d \cdot 2r_3^2 = 4 \\ f(3) = 9 &\quad \Rightarrow \quad a \cdot r_1^3 + b \cdot r_2^3 + c \cdot r_3^3 + d \cdot 3r_3^3 = 9 \end{aligned}$$

All the nasty r_i^j things are actually just constants. Solving this system gives values for a , b , c , and d that define a solution to the recurrence consistent with the boundary conditions.

3.1 An Example

Suppose that there is a type of plant that lives forever, but only reproduces during its first year of life. How fast will the plant population grow? Notice that this is just the reverse of the graduate student job problem where faculty “reproduce” in every year except the first.

Let $f(n)$ be the number of plants in year n . As boundary conditions, define $f(0) = 0$ and $f(1) = 1$. Now the plant population in year n is equal to the population from the year

before plus the number of new plants. The population from the year before is $f(n-1)$. And the number of new plants this year is equal to the number of new plants last year, which is $f(n-1) - f(n-2)$. Putting these observations together, we can form a recurrence equation for the plant population:

$$\begin{aligned} f(n) &= f(n-1) + (f(n-1) - f(n-2)) \\ &= 2f(n-1) - f(n-2) \end{aligned}$$

The characteristic equation is $x^2 - 2x + 1 = 0$, which has the single root $x = 1$ with multiplicity 2. Therefore, the solution to the recurrence has the form:

$$\begin{aligned} f(n) &= c_1(1)^n + c_2n(1)^n \\ &= c_1 + c_2n \end{aligned}$$

The boundary conditions imply two linear equations in two unknowns:

$$\begin{aligned} f(0) = 0 &\quad \Rightarrow \quad c_1 + c_2(0) = 0 \\ f(1) = 1 &\quad \Rightarrow \quad c_1 + c_2(1) = 1 \end{aligned}$$

The solution to the linear system is $c_1 = 0$, $c_2 = 1$. Therefore, the solution to the recurrence is:

$$\begin{aligned} f(n) &= c_1 + c_2n \\ &= 0 + (1)n \\ &= n \end{aligned}$$

The answer turns out to be very simple! In year n , there are exactly n plants. Of course, we probably could have solved this problem more easily with guess-and-verify. But, as the Fibonacci recurrence demonstrated, guessing is not always so easy.

4 Inhomogeneous Recurrences

We can now solve all recurrences of the form:

$$f(n) = a_1f(n-1) + a_2f(n-2) + \dots + a_df(n-d)$$

Strictly speaking, this is the family of *homogeneous* linear recurrences. Adding an extra, arbitrary function $g(n)$ on the right side gives the general form of an *inhomogeneous linear recurrence*:

$$f(n) = a_1f(n-1) + a_2f(n-2) + \dots + a_df(n-d) + g(n)$$

For example, adding $+1$ to the Fibonacci recurrence gives an inhomogeneous linear recurrence:

$$f(n) = f(n-1) + f(n-2) + 1$$

Solving inhomogeneous linear recurrences is neither very different nor very difficult. We can divide the whole job into three steps.

1. Replace $g(n)$ by 0 and solve the resulting homogeneous recurrence as before. (Ignore boundary conditions for now; that is, do not solve for constants c_1, c_2, \dots, c_d yet.) The solution to the homogeneous recurrence is called the *homogeneous solution*.
2. Now restore $g(n)$ and find a single solution to the recurrence, again ignoring boundary conditions. This is called the *particular solution*. There are general methods for finding particular solutions, but we advise you to use guess-and-verify. In a moment, we'll explain how to guess wisely.
3. Add the homogeneous and particular solutions together to obtain the *general solution*. Now use the boundary conditions to determine constants by the usual method of generating and solving a system of linear equations.

If you've studied differential equations, then all this probably sounds quite familiar. If you haven't, then— when you *do* get around to studying differential equations— *they* should seem quite familiar.

4.1 An Example

Let's demonstrate the method for solving an inhomogeneous linear recurrence on this example:

$$\begin{aligned} f(1) &= 1 \\ f(n) &= 4f(n-1) + 3^n \end{aligned}$$

Step 1: Solve the Homogeneous Recurrence

The homogeneous recurrence is $f(n) = 4f(n-1)$. The characteristic equation is $x - 4 = 0$. The only root is $x = 4$. Therefore, the homogeneous solution is $f(n) = c4^n$.

Step 2: Find a Particular Solution

Now we must find a single solution to the full recurrence $f(n) = 4f(n-1) + 3^n$. Let's guess that there is a solution of the form $d3^n$, where d is a constant. Substituting this guess into the recurrence gives:

$$\begin{aligned} d3^n &= 4d3^{n-1} + 3^n \\ 3d &= 4d + 3 \\ d &= -3 \end{aligned}$$

Evidently, $f(n) = -3 \cdot 3^n = -3^{n+1}$ is a particular solution.

Step 3: Add Solutions and Find Constants

We now add the homogeneous solution and the particular solution to obtain the general solution:

$$f(n) = c4^n - 3^{n+1}$$

The boundary condition gives the value of the constant c :

$$\begin{aligned} f(1) = 1 &\Rightarrow c4^1 - 3^{1+1} = 1 \\ &\Rightarrow c = \frac{5}{2} \end{aligned}$$

Therefore, the solution to the recurrence is $f(n) = \frac{5}{2}4^n - 3^{n+1}$. Piece of cake!

Since we could easily have made a mistake, let's check to make sure that our solution at least works for $n = 2$. From the recurrence, $f(2) = 4f(1) + 3^2 = 13$. From our closed form, $f(2) = \frac{5}{2}4^2 - 3^3 = 40 - 27 = 13$. It looks right!

4.2 How to Guess a Particular Solution

The hardest part of solving inhomogeneous recurrences is finding a particular solution. This involves guessing, and you might guess wrong. However, some rules of thumb make this job fairly easy most of the time.

- Generally, look for a particular solution with the same form as the inhomogeneous term $g(n)$.
- If $g(n)$ is a constant, then guess a particular solution $f(n) = c$. If this doesn't work, try $f(n) = bn + c$, then $f(n) = an^2 + bn + c$, etc.
- More generally, if $g(n)$ is a polynomial, try a polynomial of the same degree, then a polynomial of degree one higher, then two higher, etc. For example, if $g(n) = 6n + 5$, then try $f(n) = bn + c$ and then $f(n) = an^2 + bn + c$.
- If $g(n)$ is an exponential, such as 3^n , then first guess that $f(n) = c3^n$. Failing that, try $f(n) = bn3^n + c3^n$ and then $an^23^n + bn3^n + c3^n$, etc.

Short Guide to Solving Linear Recurrences

A *linear recurrence* is an equation

$$f(n) = \underbrace{a_1 f(n-1) + a_2 f(n-2) + \dots + a_d f(n-d)}_{\text{homogeneous part}} + \underbrace{g(n)}_{\text{inhomogeneous part}}$$

together with boundary conditions such as $f(0) = b_0$, $f(1) = b_1$, etc.

1. Find the roots of the *characteristic equation*:

$$x^n = a_1 x^{n-1} + a_2 x^{n-2} + \dots + a_k$$

2. Write down the *homogeneous solution*. Each root generates one term and the homogeneous solution is the sum of these terms. A nonrepeated root r generates the term $c_r r^n$, where c_r is a constant to be determined later. A root r with multiplicity k generates the terms:

$$c_{r_1} r_1^n, \quad c_{r_2} n r_2^n, \quad c_{r_3} n^2 r_3^n, \quad \dots, \quad c_{r_k} n^{k-1} r_k^n$$

where c_{r_1}, \dots, c_{r_k} are constants to be determined later.

3. Find a *particular solution*. This is a solution to the full recurrence that need not be consistent with the boundary conditions. Use guess and verify. If $g(n)$ is a polynomial, try a polynomial of the same degree, then a polynomial of degree one higher, then two higher, etc. For example, if $g(n) = n$, then try $f(n) = bn + c$ and then $f(n) = an^2 + bn + c$. If $g(n)$ is an exponential, such as 3^n , then first guess that $f(n) = c3^n$. Failing that, try $f(n) = bn3^n + c3^n$ and then $an^2 3^n + bn3^n + c3^n$, etc.
4. Form the *general solution*, which is the sum of the homogeneous solution and the particular solution. Here is a typical general solution:

$$f(n) = \underbrace{c2^n + d(-1)^n}_{\text{homogeneous solution}} + \underbrace{3n + 1}_{\text{particular solution}}$$

5. Substitute the boundary conditions into the general solution. Each boundary condition gives a linear equation in the unknown constants. For example, substituting $f(1) = 2$ into the general solution above gives:

$$\begin{aligned} 2 &= c \cdot 2^1 + d \cdot (-1)^1 + 3 \cdot 1 + 1 \\ \Rightarrow -2 &= 2c - d \end{aligned}$$

Determine the values of these constants by solving the resulting system of linear equations.