Illinois Institute of Technology
Department of Computer Science

Lecture 2: January 13, 2016

CS 430 Introduction to Algorithms
Spring Semester, 2016

1 Solving Recurrences

1.1 Sequences, sequence operators, and annihilators

In order to solve recurrences, we first need to understand *operations* on infinite sequences of numbers. Although these sequences are formally defined as *functions* of mapping the natural numbers to the real numbers, we will write them either as $A = \langle a_0, a_1, a_2, a_3, a_4, \ldots \rangle$ when we want to emphasize the entire sequence[1], or as $A = \langle a_i \rangle$ when we want to emphasize a generic element. For example, the Fibonacci sequence is $\langle 0, 1, 1, 2, 3, 5, 8, 13, 21, \ldots \rangle$.

We can naturally define several sequence operators:

- We can add or subtract any two sequences:

$$\langle a_i \rangle + \langle b_i \rangle = \langle a_0, a_1, a_2, \ldots \rangle + \langle b_0, b_1, b_2, \ldots \rangle = \langle a_0 + b_0, a_1 + b_1, a_2 + b_2, \ldots \rangle = \langle a_i + b_i \rangle$$

$$\langle a_i \rangle - \langle b_i \rangle = \langle a_0, a_1, a_2, \ldots \rangle - \langle b_0, b_1, b_2, \ldots \rangle = \langle a_0 - b_0, a_1 - b_1, a_2 - b_2, \ldots \rangle = \langle a_i - b_i \rangle$$

- We can multiply any sequence by a constant:

$$c \cdot \langle a_i \rangle = c \cdot \langle a_0, a_1, a_2, \ldots \rangle = \langle c \cdot a_0, c \cdot a_1, c \cdot a_2, \ldots \rangle = \langle c \cdot a_i \rangle$$

- We can shift any sequence to the left by removing its initial element:

$$\mathbf{E}\langle a_i \rangle = \mathbf{E}\langle a_0, a_1, a_2, a_3, \ldots \rangle = \langle a_1, a_2, a_3, a_4, \ldots \rangle = \langle a_{i+1} \rangle$$

Example 1. *We can understand these operators better by looking at some specific examples, using the sequence T of powers of two.*

$$\begin{aligned} T &= \langle 2^0, 2^1, 2^2, 2^3, \ldots \rangle = \langle 2^i \rangle \\ \mathbf{E}T &= \langle 2^1, 2^2, 2^3, 2^4, \ldots \rangle = \langle 2^{i+1} \rangle \\ 2T &= \langle 2 \cdot 2^0, 2 \cdot 2^1, 2 \cdot 2^2, 2 \cdot 2^3, \ldots \rangle = \langle 2^1, 2^2, 2^3, 2^4, \ldots \rangle = \langle 2^{i+1} \rangle \\ 2T - \mathbf{E}T &= \langle 2^1 - 2^1, 2^2 - 2^2, 2^3 - 2^3, 2^4 - 2^4, \ldots \rangle = \langle 0, 0, 0, 0, \ldots \rangle = \langle 0 \rangle \end{aligned}$$

1.1.1 Properties of operators

It turns out that the distributive property holds for these operators, so we can rewrite $\mathbf{E}T - 2T$ as $(\mathbf{E} - 2)T$. Since $(\mathbf{E} - 2)T = \langle 0, 0, 0, 0, \ldots \rangle$, we say that the operator $(\mathbf{E} - 2)$ *annihilates* T, and we call $(\mathbf{E} - 2)$ an

[1] It really doesn't matter whether we start a sequence with a_0 or a_1 or a_5 or even a_{-17}. Zero is often a convenient starting point for many recursively defined sequences, so we'll usually start there.

annihilator of T. Obviously, we can trivially annihilate any sequence by multiplying it by zero, so as a technical matter, we do not consider multiplication by 0 to be an annihilator.

What happens when we apply the operator $(E-3)$ to our sequence T?

$$(\mathbf{E}-3)T = \mathbf{E}T - 3T = \langle 2^{i+1}\rangle - 3\langle 2^i\rangle = \langle 2^{i+1} - 3\cdot 2^i\rangle = \langle -2^i\rangle = -T$$

The operator $(\mathbf{E}-3)$ did very little to our sequence T; it just flipped the sign of each number in the sequence. In fact, we will soon see that *only* $(\mathbf{E}-2)$ will annihilate T, and all other simple operators will affect T in very minor ways. Thus, if we know how to annihilate the sequence, we know what the sequence must look like.

In general, $(\mathbf{E}-c)$ annihilates any geometric sequence $A = \langle a_0, a_0c, a_0c^2, a_0c^3, \ldots\rangle = \langle a_0c^i\rangle$:

$$(\mathbf{E}-c)\langle a_0c^i\rangle = \mathbf{E}\langle a_0c^i\rangle - c\langle a_0c^i\rangle = \langle a_0c^{i+1}\rangle - \langle c\cdot a_0c^i\rangle = \langle a_0c^{i+1} - a_0c^{i+1}\rangle = \langle 0\rangle$$

To see that this is the only operator of this form that annihilates A, let's see the effect of operator $(\mathbf{E}-d)$ for some $d \neq c$:

$$(\mathbf{E}-d)\langle a_0c^i\rangle = \mathbf{E}\langle a_0c^i\rangle - d\langle a_0c^i\rangle = \langle a_0c^{i+1} - da_0c^i\rangle = \langle (c-d)a_0c^i\rangle = (c-d)\langle a_0c^i\rangle$$

So we have a more rigorous confirmation that an annihilator annihilates exactly one type of sequence, but multiplies other similar sequences by a constant.

We can use this fact about annihilators of geometric sequences to solve certain recurrences. For example, consider the sequence $R = \langle r_0, r_1, r_2, \ldots\rangle$ defined recursively as follows:

$$\begin{aligned} r_0 &= 3 \\ r_{i+1} &= 5r_i \end{aligned}$$

We can easily prove that the operator $(\mathbf{E}-5)$ annihilates R:

$$(\mathbf{E}-5)\langle r_i\rangle = \mathbf{E}\langle r_i\rangle - 5\langle r_i\rangle = \langle r_{i+1}\rangle - \langle 5r_i\rangle = \langle r_{i+1} - 5r_i\rangle = \langle 0\rangle$$

Since $(\mathbf{E}-5)$ is an annihilator for R, we must have the closed form solution $r_i = r_0 5^i = 3\cdot 5^i$. We can easily verify this by induction, as follows:

$$\begin{aligned} r_0 &= 3\cdot 5^0 = 3 && \text{[definition]} \\ r_i &= 5r_{i-1} && \text{[definition]} \\ &= 5\cdot(3\cdot 5^{i-1}) && \text{[induction hypothesis]} \\ &= 5^i\cdot 3 && \text{[algebra]} \end{aligned}$$

1.1.2 Multiple operators

An operator is a function that transforms one sequence into another. Like any other function, we can apply operators one after another to the same sequence. For example, we can multiply a sequence $\langle a_i\rangle$ by a constant d and then by a constant c, resulting in the sequence $c(d\langle a_i\rangle) = \langle c\cdot d\cdot a_i\rangle = (cd)\langle a_i\rangle$. Alternatively, we may multiply the sequence by a constant c and then shift it to the left to get $\mathbf{E}(c\langle a_i\rangle) = \mathbf{E}\langle c\cdot a_i\rangle = \langle c\cdot a_{i+1}\rangle$. This is exactly the same as applying the operators in the reverse order: $c(\mathbf{E}\langle a_i\rangle) = c\langle a_{i+1}\rangle = \langle c\cdot a_{i+1}\rangle$. We can also shift the sequence twice to the left: $\mathbf{E}(\mathbf{E}\langle a_i\rangle) = \mathbf{E}\langle a_{i+1}\rangle = \langle a_{i+2}\rangle$. We will write this in shorthand as $\mathbf{E}^2\langle a_i\rangle$. More generally, the operator $\mathbf{E}^k$ shifts a sequence k steps to the left: $\mathbf{E}^k\langle a_i\rangle = \langle a_{i+k}\rangle$.

We now have the tools to solve a whole host of recurrence problems. For example, what annihilates $C = \langle 2^i + 3^i \rangle$? Well, we know that $(\mathbf{E} - 2)$ annihilates $\langle 2^i \rangle$ while leaving $\langle 3^i \rangle$ essentially unscathed. Similarly, $(\mathbf{E} - 3)$ annihilates $\langle 3^i \rangle$ while leaving $\langle 2^i \rangle$ essentially unscathed. Thus, if we apply both operators one after the other, we see that $(\mathbf{E} - 2)(\mathbf{E} - 3)$ annihilates our sequence C.

In general, for any integers $a \neq b$, the operator $(\mathbf{E}-a)(\mathbf{E}-b)$ annihilates any sequence of the form $\langle c_1 a^i + c_2 b^i \rangle$ but nothing else. We will often 'multiply out' the operators into the shorthand notation $\mathbf{E}^2 - (a+b)\mathbf{E} + ab$. It is left as an exhilarating exercise to the student to verify that this shorthand actually makes sense—the operators $(\mathbf{E} - a)(\mathbf{E} - b)$ and $\mathbf{E}^2 - (a+b)\mathbf{E} + ab$ have the same effect on every sequence.

We now know finally enough to solve the recurrence for Fibonacci numbers. Specifically, notice that the recurrence $F_i = F_{i-1} + F_{i-2}$ is annihilated by $\mathbf{E}^2 - \mathbf{E} - 1$:

$$\begin{aligned}(\mathbf{E}^2 - \mathbf{E} - 1)\langle F_i \rangle &= \mathbf{E}^2\langle F_i \rangle - \mathbf{E}\langle F_i \rangle - \langle F_i \rangle \\ &= \langle F_{i+2} \rangle - \langle F_{i+1} \rangle - \langle F_i \rangle \\ &= \langle F_{i+2} - F_{i+1} - F_i \rangle \\ &= \langle 0 \rangle\end{aligned}$$

Factoring $\mathbf{E}^2 - \mathbf{E} - 1$ using the quadratic formula, we obtain

$$\mathbf{E}^2 - \mathbf{E} - 1 = (\mathbf{E} - \phi)(\mathbf{E} - \hat{\phi})$$

where $\phi = (1 + \sqrt{5})/2 \approx 1.618034$ is the golden ratio and $\hat{\phi} = (1 - \sqrt{5})/2 = 1 - \phi = -1/\phi$. Thus, the operator $(E - \phi)(E - \hat{\phi})$ annihilates the Fibonacci sequence, so F_i must have the form

$$F_i = c\phi^i + \hat{c}\hat{\phi}^i$$

for some constants c and $\hat{c}$. We call this the *generic solution* to the recurrence, since it doesn't depend at all on the base cases. To compute the constants c and $\hat{c}$, we use the base cases $F_0 = 0$ and $F_1 = 1$ to obtain a pair of linear equations:

$$\begin{aligned}F_0 = 0 &= c + \hat{c} \\ F_1 = 1 &= c\phi + \hat{c}\hat{\phi}\end{aligned}$$

Solving this system of equations gives us $c = 1/(2\phi - 1) = 1/\sqrt{5}$ and $\hat{c} = -1/\sqrt{5}$.

We now have a closed-form expression for the ith Fibonacci number:

$$\boxed{F_i = \frac{\phi^i - \hat{\phi}^i}{\sqrt{5}} = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^i - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^i}$$

With all the square roots in this formula, it's quite amazing that Fibonacci numbers are integers. However, if we do all the math correctly, all the square roots cancel out when i is an integer. (In fact, this is pretty easy to prove using the binomial theorem.)

1.1.3 Degenerate cases

We can't quite solve *every* recurrence yet. In our above formulation of $(\mathbf{E} - a)(\mathbf{E} - b)$, we assumed that $a \neq b$. What about the operator $(\mathbf{E} - a)(\mathbf{E} - a) = (\mathbf{E} - a)^2$? It turns out that this operator annihilates

sequences such as $\langle ia^i \rangle$:

$$\begin{aligned}(\mathbf{E}-a)\langle ia^i\rangle &= \langle (i+1)a^{i+1} - (a)ia^i\rangle \\ &= \langle (i+1)a^{i+1} - ia^{i+1}\rangle \\ &= \langle a^{i+1}\rangle \\ (\mathbf{E}-a)^2\langle ia^i\rangle &= (\mathbf{E}-a)\langle a^{i+1}\rangle = \langle 0\rangle\end{aligned}$$

More generally, the operator $(\mathbf{E}-a)^k$ annihilates any sequence $\langle p(i)\cdot a^i\rangle$, where $p(i)$ is any polynomial in i of degree $k-1$. As an example, $(\mathbf{E}-1)^3$ annihilates the sequence $\langle i^2\cdot 1^i\rangle = \langle i^2\rangle = \langle 1,4,9,16,25,\ldots\rangle$, since $p(i)=i^2$ is a polynomial of degree $n-1=2$.

As a review, try to explain the following statements:

- $(\mathbf{E}-1)$ annihilates any constant sequence $\langle\alpha\rangle$.
- $(\mathbf{E}-1)^2$ annihilates any arithmetic sequence $\langle\alpha+\beta i\rangle$.
- $(\mathbf{E}-1)^3$ annihilates any quadratic sequence $\langle\alpha+\beta i+\gamma i^2\rangle$.
- $(\mathbf{E}-3)(\mathbf{E}-2)(\mathbf{E}-1)$ annihilates any sequence $\langle\alpha+\beta 2^i+\gamma 3^i\rangle$.
- $(\mathbf{E}-3)^2(\mathbf{E}-2)(\mathbf{E}-1)$ annihilates any sequence $\langle\alpha+\beta 2^i+\gamma 3^i+\delta i3^i\rangle$.

1.1.4 Summary

In summary, we have learned several operators that act on sequences, as well as a few ways of combining operators.

Operator	Definition
Addition	$\langle a_i\rangle + \langle b_i\rangle = \langle a_i + b_i\rangle$
Subtraction	$\langle a_i\rangle - \langle b_i\rangle = \langle a_i - b_i\rangle$
Scalar multiplication	$c\langle a_i\rangle = \langle ca_i\rangle$
Shift	$\mathbf{E}\langle a_i\rangle = \langle a_{i+1}\rangle$
Composition of operators	$(\mathbf{X}+\mathbf{Y})\langle a_i\rangle = \mathbf{X}\langle a_i\rangle + \mathbf{Y}\langle a_i\rangle$
	$(\mathbf{X}-\mathbf{Y})\langle a_i\rangle = \mathbf{X}\langle a_i\rangle - \mathbf{Y}\langle a_i\rangle$
	$\mathbf{XY}\langle a_i\rangle = \mathbf{X}(\mathbf{Y}\langle a_i\rangle) = \mathbf{Y}(\mathbf{X}\langle a_i\rangle)$
k-fold shift	$\mathbf{E}^k\langle a_i\rangle = \langle a_{i+k}\rangle$

Notice that we have not defined a multiplication operator for two sequences. This is usually accomplished by *convolution*:

$$\langle a_i\rangle * \langle b_i\rangle = \Big\langle \sum_{j=0}^{i} a_j b_{i-j}\Big\rangle.$$

Fortunately, convolution is unnecessary for solving the recurrences we will see in this course.

We have also learned some things about annihilators, which can be summarized as follows:

Sequence	Annihilator
$\langle \alpha \rangle$	$\mathbf{E} - 1$
$\langle \alpha a^i \rangle$	$\mathbf{E} - a$
$\langle \alpha a^i + \beta b^i \rangle$	$(\mathbf{E} - a)(\mathbf{E} - b)$
$\langle \alpha_0 a_0^i + \alpha_1 a_1^i + \cdots + \alpha_n a_n^i \rangle$	$(\mathbf{E} - a_0)(\mathbf{E} - a_1) \cdots (\mathbf{E} - a_n)$
$\langle \alpha i + \beta \rangle$	$(\mathbf{E} - 1)^2$
$\langle (\alpha i + \beta) a^i \rangle$	$(\mathbf{E} - a)^2$
$\langle (\alpha i + \beta) a^i + \gamma b^i \rangle$	$(\mathbf{E} - a)^2 (\mathbf{E} - b)$
$\langle (\alpha_0 + \alpha_1 i + \cdots \alpha_{n-1} i^{n-1}) a^i \rangle$	$(\mathbf{E} - a)^n$
If $\mathbf{X}$ annihilates $\langle a_i \rangle$, then $\mathbf{X}$ also annihilates $c\langle a_i \rangle$ for any constant c.	
If $\mathbf{X}$ annihilates $\langle a_i \rangle$ and $\mathbf{Y}$ annihilates $\langle b_i \rangle$, then $\mathbf{XY}$ annihilates $\langle a_i \rangle \pm \langle b_i \rangle$.	

1.2 Solving Linear Recurrences

1.2.1 Homogeneous Recurrences

The general expressions in the annihilator box above are really the most important things to remember about annihilators because they help you to solve any recurrence for which you can write down an annihilator. The general method is:

1. Write down the annihilator for the recurrence
2. Factor the annihilator
3. Determine the sequence annihilated by each factor
4. Add these sequences together to form the generic solution
5. Solve for constants of the solution by using initial conditions

Example 2. *Let's show the steps required to solve the following recurrence:*

$$\begin{aligned} r_0 &= 1 \\ r_1 &= 5 \\ r_2 &= 17 \\ r_i &= 7r_{i-1} - 16r_{i-2} + 12r_{i-3} \end{aligned}$$

1. *Write down the annihilator. Since* $r_{i+3} - 7r_{i+2} + 16r_{i+1} - 12r_i = 0$*, the annihilator is* $\mathbf{E}^3 - 7\mathbf{E}^2 + 16\mathbf{E} - 12$.
2. *Factor the annihilator.* $\mathbf{E}^3 - 7\mathbf{E}^2 + 16\mathbf{E} - 12 = (\mathbf{E} - 2)^2(\mathbf{E} - 3)$.
3. *Determine sequences annihilated by each factor.* $(\mathbf{E} - 2)^2$ *annihilates* $\langle (\alpha i + \beta) 2^i \rangle$ *for any constants* α *and* β*, and* $(\mathbf{E} - 3)$ *annihilates* $\langle \gamma 3^i \rangle$ *for any constant* γ.
4. *Combine the sequences.* $(\mathbf{E} - 2)^2(\mathbf{E} - 3)$ *annihilates* $\langle (\alpha i + \beta) 2^i + \gamma 3^i \rangle$ *for any constants* α, β, γ.
5. *Solve for the constants. The base cases give us three equations in the three unknowns* α, β, γ*:*

$$\begin{aligned} r_0 = 1 &= (\alpha \cdot 0 + \beta) 2^0 + \gamma \cdot 3^0 = \beta + \gamma \\ r_1 = 5 &= (\alpha \cdot 1 + \beta) 2^1 + \gamma \cdot 3^1 = 2\alpha + 2\beta + 3\gamma \\ r_2 = 17 &= (\alpha \cdot 2 + \beta) 2^2 + \gamma \cdot 3^2 = 8\alpha + 4\beta + 9\gamma \end{aligned}$$

We can solve these equations to get $\alpha = 1$, $\beta = 0$, $\gamma = 1$. Thus, our final solution is $\boxed{r_i = i2^i + 3^i}$, *which we can verify by induction.*

1.2.2 Non-homogeneous Recurrences

A *height balanced tree* is a binary tree, where the heights of the two subtrees of the root differ by at most one, and both subtrees are also height balanced. To ground the recursive definition, the empty set is considered a height balanced tree of height -1, and a single node is a height balanced tree of height 0.

Let T_n be the smallest height-balanced tree of height n—how many nodes does T_n have? Well, one of the subtrees of T_n has height $n-1$ (since T_n has height n) and the other has height either $n-1$ or $n-2$ (since T_n is height-balanced and as small as possible). Since both subtrees are themselves height-balanced, the two subtrees must be T_{n-1} and T_{n-2}.

We have just derived the following recurrence for t_n, the number of nodes in the tree T_n:

$$\begin{aligned} t_{-1} &= 0 && \text{[the empty set]} \\ t_0 &= 1 && \text{[a single node]} \\ t_n &= t_{n-1} + t_{n-2} + 1 \end{aligned}$$

The final '+1' is for the root of T_n.

We refer to the terms in the equation involving t_i's as the *homogeneous* terms and the rest as the *non-homogeneous* terms. (If there were no non-homogeneous terms, we would say that the recurrence itself is homogeneous.) We know that $\mathbf{E}^2 - \mathbf{E} - 1$ annihilates the homogeneous part $t_n = t_{n-1} + t_{n-2}$. Let us try applying this annihilator to the entire equation:

$$\begin{aligned} (\mathbf{E}^2 - \mathbf{E} - 1)\langle t_i \rangle &= \mathbf{E}^2\langle t_i \rangle - \mathbf{E}\langle t_i \rangle - 1\langle t_i \rangle \\ &= \langle t_{i+2} \rangle - \langle t_{i+1} \rangle - \langle t_i \rangle \\ &= \langle t_{i+2} - t_{i+1} - t_i \rangle \\ &= \langle 1 \rangle \end{aligned}$$

The leftover sequence $\langle 1, 1, 1, \ldots \rangle$ is called the *residue*. To obtain the annihilator for the entire recurrence, we compose the annihilator for its homogeneous part with the annihilator of its residue. Since $\mathbf{E} - 1$ annihilates $\langle 1 \rangle$, it follows that $(\mathbf{E}^2 - \mathbf{E} - 1)(\mathbf{E} - 1)$ annihilates $\langle t_n \rangle$. We can factor the annihilator into

$$(\mathbf{E} - \phi)(\mathbf{E} - \hat{\phi})(\mathbf{E} - 1),$$

so our annihilator rules tell us that

$$t_n = \alpha\phi^n + \beta\hat{\phi}^n + \gamma$$

for some constants α, β, γ. We call this the *generic solution* to the recurrence. Different recurrences can have the same generic solution.

To solve for the unknown constants, we need three equations in three unknowns. Our base cases give us two equations, and we can get a third by examining the next nontrivial case $t_1 = 2$:

$$\begin{aligned} t_{-1} = 0 &= \alpha\phi^{-1} + \beta\hat{\phi}^{-1} + \gamma = \alpha/\phi + \beta/\hat{\phi} + \gamma \\ t_0 = 1 &= \alpha\phi^0 + \beta\hat{\phi}^0 + \gamma = \alpha + \beta + \gamma \\ t_1 = 2 &= \alpha\phi^1 + \beta\hat{\phi}^1 + \gamma = \alpha\phi + \beta\hat{\phi} + \gamma \end{aligned}$$

Solving these equations, we find that $\alpha = \frac{\sqrt{5}+2}{\sqrt{5}}$, $\beta = \frac{\sqrt{5}-2}{\sqrt{5}}$, and $\gamma = -1$. Thus,

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

Here is the general method for non-homogeneous recurrences:

1. Write down the homogeneous annihilator, directly from the recurrence
1½. 'Multiply' by the annihilator for the residue
2. Factor the annihilator
3. Determine what sequence each factor annihilates
4. Add these sequences together to form the generic solution
5. Solve for constants of the solution by using initial conditions

1.2.3 Some more examples

In each example below, we use the base cases $a_0 = 0$ and $a_1 = 1$.

- $\boldsymbol{a_n = a_{n-1} + a_{n-2} + 2}$
  - The homogeneous annihilator is $\mathbf{E}^2 - \mathbf{E} - 1$.
  - The residue is the constant sequence $\langle 2, 2, 2, \ldots \rangle$, which is annihilated by $\mathbf{E} - 1$.
  - Thus, the annihilator is $(\mathbf{E}^2 - \mathbf{E} - 1)(\mathbf{E} - 1)$.
  - The annihilator factors into $(\mathbf{E} - \phi)(\mathbf{E} - \hat{\phi})(\mathbf{E} - 1)$.
  - Thus, the generic solution is $a_n = \alpha\phi^n + \beta\hat{\phi}^n + \gamma$.
  - The constants α, β, γ satisfy the equations

$$\begin{aligned} a_0 = 0 &= \alpha + \beta + \gamma \\ a_1 = 1 &= \alpha\phi + \beta\hat{\phi} + \gamma \\ a_2 = 3 &= \alpha\phi^2 + \beta\hat{\phi}^2 + \gamma \end{aligned}$$

  - Solving the equations gives us $\alpha = \frac{\sqrt{5}+2}{\sqrt{5}}$, $\beta = \frac{\sqrt{5}-2}{\sqrt{5}}$, and $\gamma = -2$
  - So the final solution is $\boxed{a_n = \frac{\sqrt{5}+2}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n + \frac{\sqrt{5}-2}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n - 2}$

  (In the remaining examples, I won't explicitly enumerate the steps like this.)

- $\boldsymbol{a_n = a_{n-1} + a_{n-2} + 3}$

  The homogeneous annihilator $(\mathbf{E}^2 - \mathbf{E} - 1)$ leaves a constant residue $\langle 3, 3, 3, \ldots \rangle$, so the annihilator is $(\mathbf{E}^2 - \mathbf{E} - 1)(\mathbf{E} - 1)$, and the generic solution is $a_n = \alpha\phi^n + \beta\hat{\phi}^n + \gamma$. Solving the equations

$$\begin{aligned} a_0 = 0 &= \alpha + \beta + \gamma \\ a_1 = 1 &= \alpha\phi + \beta\hat{\phi} + \gamma \\ a_2 = 4 &= \alpha\phi^2 + \beta\hat{\phi}^2 + \gamma \end{aligned}$$

  gives us the final solution $\boxed{a_n = \frac{\sqrt{5}+3}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n + \frac{\sqrt{5}-3}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n - 3}$

- $\boldsymbol{a_n = a_{n-1} + a_{n-2} + 2^n}$

  The homogeneous annihilator $(\mathbf{E}^2 - \mathbf{E} - 1)$ leaves an exponential residue $\langle 4, 8, 16, 32, \ldots \rangle = \langle 2^{i+2} \rangle$, which is annihilated by $\mathbf{E} - 2$. Thus, the annihilator is $(\mathbf{E}^2 - \mathbf{E} - 1)(\mathbf{E} - 2)$, and the generic solution is $a_n = \alpha\phi^n + \beta\hat{\phi}^n + \gamma 2^n$. The constants α, β, γ satisfy the following equations:

$$\begin{aligned} a_0 = 0 &= \alpha + \beta + \gamma \\ a_1 = 1 &= \alpha\phi + \beta\hat{\phi} + 2\gamma \\ a_2 = 5 &= \alpha\phi^2 + \beta\hat{\phi}^2 + 4\gamma \end{aligned}$$

- $\boldsymbol{a_n = a_{n-1} + a_{n-2} + n}$

  The homogeneous annihilator $(\mathbf{E}^2 - \mathbf{E} - 1)$ leaves a linear residue $\langle 2, 3, 4, 5 \ldots \rangle = \langle i + 2 \rangle$, which is annihilated by $(\mathbf{E} - 1)^2$. Thus, the annihilator is $(\mathbf{E}^2 - \mathbf{E} - 1)(\mathbf{E} - 1)^2$, and the generic solution is $a_n = \alpha\phi^n + \beta\hat{\phi}^n + \gamma + \delta n$. The constants $\alpha, \beta, \gamma, \delta$ satisfy the following equations:

$$\begin{aligned} a_0 = 0 &= \alpha + \beta + \gamma \\ a_1 = 1 &= \alpha\phi + \beta\hat{\phi} + \gamma + \delta \\ a_2 = 3 &= \alpha\phi^2 + \beta\hat{\phi}^2 + \gamma + 2\delta \\ a_3 = 7 &= \alpha\phi^3 + \beta\hat{\phi}^3 + \gamma + 3\delta \end{aligned}$$

- $\boldsymbol{a_n = a_{n-1} + a_{n-2} + n^2}$

  The homogeneous annihilator $(\mathbf{E}^2 - \mathbf{E} - 1)$ leaves a quadratic residue $\langle 4, 9, 16, 25 \ldots \rangle = \langle (i + 2)^2 \rangle$, which is annihilated by $(\mathbf{E} - 1)^3$. Thus, the annihilator is $(\mathbf{E}^2 - \mathbf{E} - 1)(\mathbf{E} - 1)^3$, and the generic solution is $a_n = \alpha\phi^n + \beta\hat{\phi}^n + \gamma + \delta n + \epsilon n^2$. The constants $\alpha, \beta, \gamma, \delta, \epsilon$ satisfy the following equations:

$$\begin{aligned} a_0 = 0 &= \alpha + \beta + \gamma \\ a_1 = 1 &= \alpha\phi + \beta\hat{\phi} + \gamma + \delta + \epsilon \\ a_2 = 5 &= \alpha\phi^2 + \beta\hat{\phi}^2 + \gamma + 2\delta + 4\epsilon \\ a_3 = 15 &= \alpha\phi^3 + \beta\hat{\phi}^3 + \gamma + 3\delta + 9\epsilon \\ a_4 = 36 &= \alpha\phi^4 + \beta\hat{\phi}^4 + \gamma + 4\delta + 16\epsilon \end{aligned}$$

- $\boldsymbol{a_n = a_{n-1} + a_{n-2} + n^2 - 2^n}$

  The homogeneous annihilator $(\mathbf{E}^2 - \mathbf{E} - 1)$ leaves the residue $\langle (i + 2)^2 - 2^{i-2} \rangle$. The quadratic part of the residue is annihilated by $(\mathbf{E} - 1)^3$, and the exponential part is annihilated by $(\mathbf{E} - 2)$. Thus, the annihilator for the whole recurrence is $(\mathbf{E}^2 - \mathbf{E} - 1)(\mathbf{E} - 1)^3(\mathbf{E} - 2)$, and so the generic solution is $a_n = \alpha\phi^n + \beta\hat{\phi}^n + \gamma + \delta n + \epsilon n^2 + \eta 2^n$. The constants $\alpha, \beta, \gamma, \delta, \epsilon, \eta$ satisfy a system of six equations in six unknowns determined by $a_0, a_1, \ldots, a_5$.

- $\boldsymbol{a_n = a_{n-1} + a_{n-2} + \phi^n}$

  The annihilator is $(\mathbf{E}^2 - \mathbf{E} - 1)(\mathbf{E} - \phi) = (\mathbf{E} - \phi)^2(\mathbf{E} - \hat{\phi})$, so the generic solution is $a_n = \alpha\phi^n + \beta n\phi^n + \gamma\hat{\phi}^n$. (Other recurrence solving methods will have a "interference" problem with this equation, while the operator method does not.)

Our method does not work on recurrences like $a_n = a_{n-1} + \frac{1}{n}$ or $a_n = a_{n-1} + \lg n$, because the functions $\frac{1}{n}$ and $\lg n$ do not have annihilators. Our tool, as it stands, is limited to linear recurrences.

1.3 Divide and Conquer Recurrences

Divide and conquer algorithms often give us running-time recurrences of the form

$$T(n) = aT(n/b) + f(n) \tag{1}$$

where a and b are constants and $f(n)$ is some other function. The so-called 'Master Theorem' gives us a general method for solving such recurrences when $f(n)$ is a simple polynomial.

Unfortunately, the Master Theorem doesn't work for all functions $f(n)$, and many useful recurrences don't look like (1) at all. Fortunately, there's a more general technique to solve most divide-and-conquer recurrences, even if they don't have this form. This technique is used to *prove* the Master Theorem, so if you remember this technique, you can forget the Master Theorem entirely (which is what I did). Throw off your chains!

We illustrate the technique using the generic recurrence (1). We start by drawing a *recursion tree*, shown in Figure 1. The root of the recursion tree is a box containing the value $f(n)$, it has a children, each of which is the root of a recursion tree for $T(n/b)$. Equivalently, a recursion tree is a complete a-ary tree where each node at depth i contains the value $a^i f(n/b^i)$. The recursion stops when we get to the base case(s) of the recurrence. Since we're looking for asymptotic bounds, it turns out not to matter much what we use for the base case; for purposes of illustration, assume that $T(1) = f(1)$.

Now $T(n)$ is just the sum of all values stored in the tree. Assuming that each level of the tree is full, we have

$$T(n) = f(n) + a\,f(n/b) + a^2\,f(n/b^2) + \cdots + a^i\,f(n/b^i) + \cdots + a^L\,f(n/b^L)$$

where L is the depth of the recursion tree. We easily see that $L = \log_b n$, since $n/b^L = 1$. Since $f(1) = \Theta(1)$, the last non-zero term in the summation is $\Theta(a^L) = \Theta(a^{\log_b n}) = \Theta(n^{\log_b a})$.

Now we can easily state and prove the Master Theorem, in a slightly different form than it's usually stated.

The Master Theorem. The recurrence $T(n) = aT(n/b) + f(n)$ can be solved as follows.

- If $af(n/b)/f(n) < 1$, then $T(n) = \Theta(f(n))$.
- If $af(n/b)/f(n) > 1$, then $T(n) = \Theta(n^{\log_b a})$.
- If $af(n/b)/f(n) = 1$, then $T(n) = \Theta(f(n)\log_b n)$.
- If none of these three cases apply, you're on your own.

Proof. If $f(n)$ is a *constant factor larger* than $af(n/b)$, then by induction, the sum is a descending geometric series. The sum of any geometric series is a constant times its largest term. In this case, the largest term is the first term $f(n)$.

If $f(n)$ is a *constant factor smaller* than $af(n/b)$, then by induction, the sum is an ascending geometric series. The sum of any geometric series is a constant times its largest term. In this case, this is the last term, which by our earlier argument is $\Theta(n^{\log_b a})$.

Finally, if $af(n/b) = f(n)$, then by induction, each of the $L+1$ terms in the summation is equal to $f(n)$. □

Here are a few canonical examples of the Master Theorem in action:

- **Randomized selection: $T(n) = T(3n/4) + n$**

  Here $a\,f(n/b) = 3n/4$ is smaller than $f(n) = n$ by a factor of $4/3$, so $\boxed{T(n) = \Theta(n)}$

- **Karatsuba's multiplication algorithm: $T(n) = 3T(n/2) + n$**

  Here $a\,f(n/b) = 3n/2$ is bigger than $f(n) = n$ by a factor of $3/2$, so $\boxed{T(n) = \Theta(n^{\log_2 3})}$

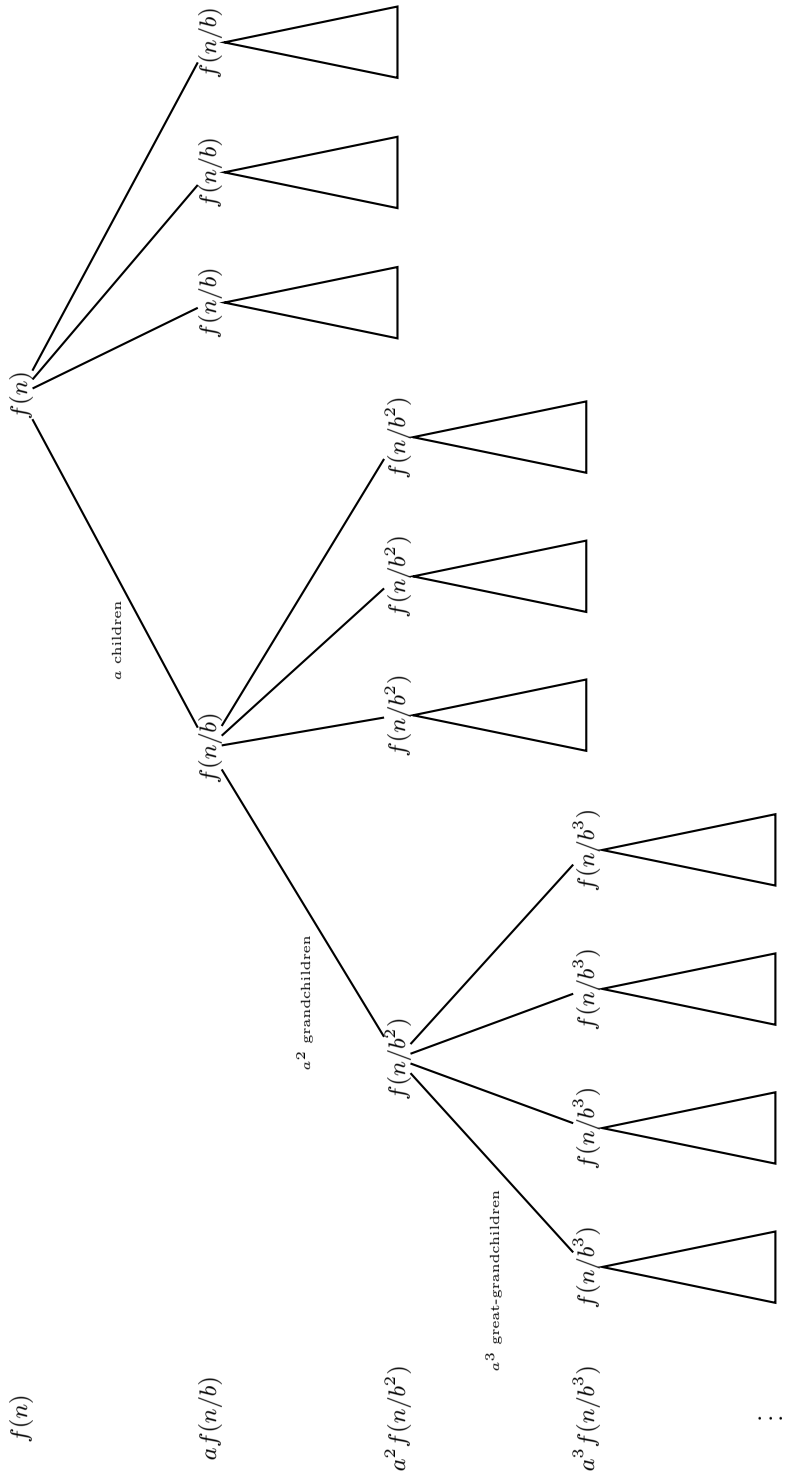

Figure 1: A recursion tree for the recurrence $T(n) = aT(n/b) + f(n)$. The tree has height $L = \log_b n$.

- **Mergesort: $T(n) = 2T(n/2) + n$**

  Here $a\,f(n/b) = f(n)$, so $\boxed{T(n) = \Theta(n \log n)}$

- **$T(n) = 4T(n/2) + n \lg n$**

  In this case, we have $a\,f(n/b) = 2n \lg n - 2n$, which is not quite twice $f(n) = n \lg n$. However, for sufficiently large n (which is all we care about with asymptotic bounds) we have $2f(n) > a\,f(n/b) > 1.9f(n)$. Since the level sums are bounded both above and below by ascending geometric series, the solution is $T(n) = \Theta(n^{\log_2 4}) = \Theta(n^2)$. (This trick will *not* work in the first or third cases of the Master Theorem!)

Using the same recursion-tree technique, we can also solve recurrences where the Master Theorem doesn't apply.

- **$T(n) = 2T(n/2) + n/\lg n$**

  We can't apply the Master Theorem here, because $a\,f(n/b) = n/(\lg n - 1)$ isn't equal to $f(n) = n/\lg n$, but the difference isn't a constant factor. So we need to compute each of the level sums and compute their total in some other way. It's not hard to see that the sum of all the nodes in the ith level is $n/(\lg n - i)$. In particular, this means the depth of the tree is at most $\lg n - 1$.

$$T(n) = \sum_{i=0}^{\lg n - 1} \frac{n}{\lg n - i} = \sum_{j=1}^{\lg n} \frac{n}{j} = nH_{\lg n} = \boxed{\Theta(n \lg \lg n)}$$

- **Randomized quicksort: $T(n) = T(3n/4) + T(n/4) + n$**

  In this case, nodes in the same level of the recursion tree have different values. This makes the tree lopsided; different leaves are at different levels. However, it's not to hard to see that the nodes in any *complete* level (that is, above any of the leaves) sum to n, so this is like the last case of the Master Theorem, and that every leaf has depth between $\log_4 n$ and $\log_{4/3} n$. To derive an upper bound, we overestimate $T(n)$ by ignoring the base cases and extending the tree downward to the level of the *deepest* leaf. Similarly, to derive a lower bound, we overestimate $T(n)$ by counting only nodes in the tree up to the level of the *shallowest* leaf. These observations give us the upper and lower bounds $n \log_4 n \le T(n) \le n \log_{4/3} n$. Since these bounds differ by only a constant factor, we have $\boxed{T(n) = \Theta(n \log n)}$.

- **Deterministic selection: $T(n) = T(n/5) + T(7n/10) + n$**

  Again, we have a lopsided recursion tree. If we look only at complete levels of the tree, we find that the level sums form a descending geometric series $T(n) = n + 9n/10 + 81n/100 + \cdots$, so this is like the first case of the master theorem. We can get an upper bound by ignoring the base cases entirely and growing the tree out to infinity, and we can get a lower bound by only counting nodes in complete levels. Either way, the geometric series is dominated by its largest term, so $\boxed{T(n) = \Theta(n)}$.

- **$T(n) = \sqrt{n} \cdot T(\sqrt{n}) + n$**

  In this case, we have a complete recursion tree, but the *degree* of the nodes is no longer constant, so we have to be a bit more careful. It's not hard to see that the nodes in any level sum to n, so this is like the third case of the Master Theorem. The depth L satisfies the identity $n^{2^{-L}} = 2$ (we can't get all the way down to 1 by taking square roots), so $L = \lg \lg n$ and $\boxed{T(n) = \Theta(n \lg \lg n)}$.

- $\boldsymbol{T(n) = 4\sqrt{n} \cdot T(\sqrt{n}) + n}$

  We still have at most $\lg \lg n$ levels, but now the nodes in level i sum to $4^i n$. We have an increasing geometric series of level sums, like the second Master case, so $T(n)$ is dominated by the sum over the deepest level: $T(n) = \Theta(4^{\lg \lg n} n) = \boxed{\Theta(n \log^2 n)}$

1.4 Transforming Recurrences

1.4.1 An analysis of mergesort: domain transformation

Previously we gave the recurrence for mergesort as $T(n) = 2T(n/2) + n$, and obtained the solution $T(n) = \Theta(n \log n)$ using the Master Theorem (or the recursion tree method if you, like me, can't remember the Master Theorem). This is fine if n is a power of two, but for other values values of n, this recurrence is incorrect. When n is odd, then the recurrence calls for us to sort a fractional number of elements! Worse yet, if n is not a power of two, we will *never* reach the base case $T(1) = 0$.

To get a recurrence that's valid for *all* integers n, we need to carefully add ceilings and floors:

$$T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + n.$$

We have almost no hope of getting an exact solution here; the floors and ceilings will eventually kill us. So instead, let's just try to get a tight asymptotic upper bound for $T(n)$ using a technique called *domain transformation*. A domain transformation rewrites a function $T(n)$ with a difficult recurrence as a nested function $S(f(n))$, where $f(n)$ is a simple function and $S()$ has an easier recurrence.

First we overestimate the time bound, once by pretending that the two subproblem sizes are equal, and again to eliminate the ceiling:

$$T(n) \le 2T\big(\lceil n/2 \rceil\big) + n \le 2T(n/2 + 1) + n.$$

Now we define a new function $S(n) = T(n + \alpha)$, where α is a unknown constant, chosen so that $S(n)$ satisfies the Master-ready recurrence $S(n) \le 2S(n/2) + O(n)$. To figure out the correct value of α, we compare two versions of the recurrence for the function $T(n + \alpha)$:

$$\begin{array}{rcll} S(n) & \le & 2S(n/2) + O(n) & \text{implies} \quad T(n+\alpha) \le 2T(n/2 + \alpha) + O(n) \\ T(n) & \le & 2T(n/2 + 1) + n & \text{implies} \quad T(n+\alpha) \le 2T((n+\alpha)/2 + 1) + n + \alpha \end{array}$$

For these two recurrences to be equal, we need $n/2 + \alpha = (n + \alpha)/2 + 1$, which implies that $\alpha = 2$. The Master Theorem now tells us that $S(n) = O(n \log n)$, so

$$T(n) = S(n - 2) = O((n - 2) \log(n - 2)) = O(n \log n).$$

A similar argument gives a matching lower bound $T(n) = \Omega(n \log n)$. So $\boxed{T(n) = \Theta(n \log n)}$ after all, just as though we had ignored the floors and ceilings from the beginning!

Domain transformations are useful for removing floors, ceilings, and lower order terms from the arguments of any recurrence that otherwise looks like it ought to fit either the Master Theorem or the recursion tree method. But now that we know this, we don't need to bother grinding through the actual gory details!

1.4.2 A less trivial example

There is a data structure in computational geometry called *ham-sandwich trees*, where the cost of doing a certain search operation obeys the recurrence $\boldsymbol{T(n) = T(n/2) + T(n/4) + 1}$. This doesn't fit the Master

theorem, because the two subproblems have different sizes, and using the recursion tree method only gives us the loose bounds $\sqrt{n} \ll T(n) \ll n$.

Domain transformations save the day. If we define the new function $t(k) = T(2^k)$, we have a new recurrence

$$t(k) = t(k-1) + t(k-2) + 1$$

which should immediately remind you of Fibonacci numbers. Sure enough, after a bit of work, the annihilator method gives us the solution $t(k) = \Theta(\phi^k)$, where $\phi = (1+\sqrt{5})/2$ is the golden ratio. This implies that

$$T(n) = t(\lg n) = \Theta(\phi^{\lg n}) = \boxed{\Theta(n^{\lg \phi})} \approx \Theta(n^{0.69424}).$$

It's possible to solve this recurrence without domain transformations and annihilators—in fact, the inventors of ham-sandwich trees did so—but it's much more difficult.

1.4.3 Secondary recurrences

Consider the recurrence $\boldsymbol{T(n) = 2T(\frac{n}{3} - 1) + n}$ with the base case $T(1) = 1$. We already know how to use domain transformations to get the tight asymptotic bound $T(n) = \Theta(n)$, but how would we we obtain an *exact* solution?

First we need to figure out how the parameter n changes as we get deeper and deeper into the recurrence. For this we use a *secondary recurrence*. We define a sequence n_i so that

$$T(n_i) = 2T(n_{i-1}) + n_i,$$

So n_i is the argument of $T()$ when we are i recursion steps away from the base case $n_0 = 1$. The original recurrence gives us the following secondary recurrence for n_i:

$$n_{i-1} = \frac{n_i}{3} - 1 \text{ implies } n_i = 3n_{i-1} + 3.$$

The annihilator for this recurrence is $(\mathbf{E}-1)(\mathbf{E}-3)$, so the generic solution is $n_i = \alpha 3^i + \beta$. Plugging in the base cases $n_0 = 1$ and $n_1 = 6$, we get the exact solution

$$n_i = \frac{5}{2} \cdot 3^i - \frac{3}{2}.$$

Notice that our original function $T(n)$ is only well-defined if $n = n_i$ for some integer $i \geq 0$.

Now to solve the original recurrence, we do a range transformation. If we set $t_i = T(n_i)$, we have the recurrence $t_i = 2t_{i-1} + \frac{5}{2} \cdot 3^i - \frac{3}{2}$, which by now we can solve using the annihilator method. The annihilator of the recurrence is $(\mathbf{E}-2)(\mathbf{E}-3)(\mathbf{E}-1)$, so the generic solution is $\alpha' 3^i + \beta' 2^i + \gamma'$. Plugging in the base cases $t_0 = 1$, $t_1 = 8$, $t_2 = 37$, we get the exact solution

$$t_i = \frac{15}{2} \cdot 3^i - 8 \cdot 2^i + \frac{3}{2}$$

Finally, we need to substitute to get a solution for the original recurrence in terms of n, by inverting the solution of the secondary recurrence. If $n = n_i = \frac{5}{2} \cdot 3^i - \frac{3}{2}$, then (after a little algebra) we have

$$i = \log_3 \left(\frac{2}{5} n + \frac{3}{5} \right).$$

Substituting this into the expression for t_i, we get our exact, closed-form solution.

$$\begin{aligned} T(n) &= \frac{15}{2} \cdot 3^i - 8 \cdot 2^i + \frac{3}{2} \\ &= \frac{15}{2} \cdot 3^{\log_3\left(\frac{2}{5}n+\frac{3}{5}\right)} - 8 \cdot 2^{\log_3\left(\frac{2}{5}n+\frac{3}{5}\right)} + \frac{3}{2} \\ &= \frac{15}{2}\left(\frac{2}{5}n+\frac{3}{5}\right) - 8 \cdot \left(\frac{2}{5}n+\frac{3}{5}\right)^{\log_3 2} + \frac{3}{2} \\ &= 3n - 8 \cdot \left(\frac{2}{5}n+\frac{3}{5}\right)^{\log_3 2} + 6 \end{aligned}$$

Isn't that special? Now you know why we stick to asymptotic bounds for most recurrences.

1.5 The Ultimate Method: Guess and Confirm

Ultimately, there is one failsafe method to solve *any* recurrence:

Guess the answer, and then prove it correct by induction.

The annihilator method, the recursion-tree method, and transformations are good ways to generate guesses that are guaranteed to be correct, provided you use them correctly. But if you're faced with a recurrence that doesn't seem to fit any of these methods, or if you've forgotten how those techniques work, don't despair! If you guess a closed-form solution and then try to verify your guess inductively, usually either the proof succeeds and you're done, or the proof fails in a way that lets you refine your guess. Where you get your initial guess is utterly irrelevant[2]—from a classmate, from a textbook, on the web, from the answer to a different problem, scrawled on a bathroom wall in Siebel, dictated by the machine elves, whatever. If you can prove that the answer is correct, then it's correct!

1.5.1 Tower of Hanoi

The classical Tower of Hanoi problem gives us the recurrence $\boldsymbol{T(n) = 2T(n-1) + 1}$ with base case $\boldsymbol{T(0) = 0}$. Just looking at the recurrence we can guess that $T(n)$ is something like 2^n. If we write out the first few values of $T(n)$ all the values are one less than a power of two.

$$T(0) = 0,\ T(1) = 1,\ T(2) = 3,\ T(3) = 7,\ T(4) = 15,\ T(5) = 31,\ T(6) = 63, \ldots,$$

It looks like $\boxed{\boldsymbol{T(n) = 2^n - 1}}$ might be the right answer. Let's check.

$$\begin{aligned} T(0) &= 0 = 2^0 - 1 \\ T(n) &= 2T(n-1) + 1 \\ &= 2(2^{n-1} - 1) + 1 \quad \text{[induction hypothesis]} \\ &= 2^n - 1 \quad \text{[algebra]} \end{aligned}$$

We were right!

[2] . . . except of course during exams, where you aren't supposed to use *any* outside sources

1.5.2 Fibonacci numbers

Let's try a less trivial example: the Fibonacci numbers $\boldsymbol{F_n = F_{n-1} + F_{n-2}}$ with base cases $\boldsymbol{F_0 = 0}$ and $\boldsymbol{F_1 = 1}$. There is no obvious pattern (besides the obvious one) in the first several values, but we can reasonably guess that F_n is exponential in n. Let's try to prove that $F_n \le a \cdot c^n$ for some constants $a > 0$ and $c > 1$ and see how far we get.

$$F_n \;=\; F_{n-1} + F_{n-2} \;\le\; a \cdot c^{n-1} + a \cdot c^{n-2} \;\le\; a \cdot c^n \;???$$

The last inequality is satisfied if $c^n \ge c^{n-1} + c^{n-2}$, or more simply, if $c^2 - c - 1 \ge 0$. The smallest value of c that works is $\phi = (1 + \sqrt{5})/2 \approx 1.618034$; the other root of the quadratic equation is negative, so we can ignore it.

So we have *most* of an inductive proof that $F_n \le a \cdot \phi^n$ for *any* constant a. All that we're missing are the base cases, which (we can easily guess) must determine the value of the coefficient a. We quickly compute

$$\frac{F_0}{\phi^0} = 0 \quad \text{and} \quad \frac{F_1}{\phi^1} = \frac{1}{\phi} \approx 0.618034 > 0,$$

so the base cases of our induction proof are correct as long as $a \ge 1/\phi$. It follows that $\boldsymbol{F_n \le \phi^{n-1}}$ for all $n \ge 0$.

What about a matching lower bound? Well, the same inductive proof implies that $F_n \ge b \cdot \phi^n$ for some constant b, but the only value of b that works for *all* n is the trivial $b = 0$. We could try to find some lower-order term that makes the base case non-trivial, but an easier approach is to recall that $\Omega()$ bounds only have to work for sufficiently large n. So let's ignore the trivial base case $F_0 = 0$ and assume that $F_2 = 1$ is a base case instead. Some more calculation gives us

$$\frac{F_2}{\phi^2} = \frac{1}{\phi^2} \approx 0.381966 < \frac{1}{\phi}.$$

Thus, the new base cases of our induction proof are correct as long as $b \le 1/\phi^2$, which implies that $\boldsymbol{F_n \ge \phi^{n-2}}$ for all $n \ge 1$.

Putting the upper and lower bounds together, we correctly conclude that $\boxed{\boldsymbol{F_n = \Theta(\phi^n)}}$. It *is* possible to get a more exact solution by speculatively refining and conforming our current bounds, but it's not easy; you're better off just using annihilators.

1.5.3 A divide-and-conquer example

Consider the divide-and-conquer recurrence $\boldsymbol{T(n) = \sqrt{n} \cdot T(\sqrt{n}) + n}$. It doesn't fit into the form required by the Master Theorem, but it still sort of resembles the Mergesort recurrence—the total size of the subproblems at the first level of recursion is n—so let's *guess* that $T(n) = O(n \log n)$, and then try to prove that our guess is correct. Specifically, let's conjecture that $T(n) \le a\, n \lg n$ for all sufficiently large n and some constant a to be determined later:

$$\begin{aligned} T(n) &= \sqrt{n} \cdot T(\sqrt{n}) + n \\ &\le \sqrt{n} \cdot a \sqrt{n} \lg \sqrt{n} + n \quad \text{[induction hypothesis]} \\ &= (a/2) n \lg n + n \quad \text{[algebra]} \\ &\le a n \lg n \end{aligned}$$

The last inequality assumes only that $1 \le (a/2) \log n$, or equivalently, that $n \ge 2^{2/a}$. In other words, the induction proof is correct if n is sufficiently large. So we were right!

But before you break out the champagne, what about the multiplicative constant a? The proof worked for *any* constant a, no matter how small. This strongly suggests that our upper bound $T(n) = O(n \log n)$ is not tight. Indeed, if we try to prove a matching lower bound $T(n) \geq b\, n \log n$ for sufficiently large n, we run into trouble.

$$\begin{aligned} T(n) &= \sqrt{n} \cdot T(\sqrt{n}) + n \\ &\geq \sqrt{n} \cdot b\sqrt{n} \log \sqrt{n} + n \quad \text{[induction hypothesis]} \\ &= (b/2)n \log n + n \\ &\not\geq bn \log n \end{aligned}$$

The last inequality would be correct only if $1 > (b/2) \log n$, but that inequality is false for large values of n, no matter which constant b we choose. Okay, so $\Theta(n \log n)$ is too big. How about $\Theta(n)$? The lower bound is easy to prove directly:

$$T(n) = \sqrt{n} \cdot T(\sqrt{n}) + n \geq n$$

But an inductive proof of the lower bound fails.

$$\begin{aligned} T(n) &= \sqrt{n} \cdot T(\sqrt{n}) + n \\ &\leq \sqrt{n} \cdot a\sqrt{n} + n \quad \text{[induction hypothesis]} \\ &= (a+1)n \quad \text{[algebra]} \\ &\not\leq an \end{aligned}$$

Hmmm. So what's bigger than n and smaller than $n \lg n$? How about $n\sqrt{\lg n}$?

$$\begin{aligned} T(n) &= \sqrt{n} \cdot T(\sqrt{n}) + n \\ &\leq \sqrt{n} \cdot a\sqrt{n}\sqrt{\lg \sqrt{n}} + n \quad \text{[induction hypothesis]} \\ &= (a/\sqrt{2})\, n\sqrt{\lg n} + n \quad \text{[algebra]} \\ &\leq a\, n\sqrt{\lg n} \quad \text{for large enough } n \end{aligned}$$

Okay, the upper bound checks out; how about the lower bound?

$$\begin{aligned} T(n) = \sqrt{n} \cdot T(\sqrt{n}) + n &\geq \sqrt{n} \cdot b\sqrt{n}\sqrt{\lg \sqrt{n}} + n \quad \text{[induction hypothesis]} \\ &= (b/\sqrt{2})\, n\sqrt{\lg n} + n \quad \text{[algebra]} \\ &\not\geq b\, n\sqrt{\lg n} \end{aligned}$$

No, the last step doesn't work. So $\Theta(n\sqrt{\lg n})$ doesn't work. Hmmm... what else is between n and $n \lg n$? How about $n \lg \lg n$?

$$\begin{aligned} T(n) = \sqrt{n} \cdot T(\sqrt{n}) + n &\leq \sqrt{n} \cdot a\sqrt{n} \lg \lg \sqrt{n} + n \quad \text{[induction hypothesis]} \\ &= a\, n \lg \lg n - a\, n + n \quad \text{[algebra]} \\ &\leq a\, n \lg \lg n \qquad \textbf{if } \boldsymbol{a \geq 1} \end{aligned}$$

Hey look at that! For once, our upper bound proof requires a constraint on the hidden constant a. This is a good indication that we've found the right answer. Let's try the lower bound:

$$\begin{aligned} T(n) = \sqrt{n} \cdot T(\sqrt{n}) + n &\geq \sqrt{n} \cdot b\sqrt{n} \lg \lg \sqrt{n} + n \quad \text{[induction hypothesis]} \\ &= b\, n \lg \lg n - b\, n + n \quad \text{[algebra]} \\ &\geq b\, n \lg \lg n \qquad \textbf{if } \boldsymbol{b \leq 1} \end{aligned}$$

Hey, it worked! We have most of an inductive proof that $T(n) \leq an \lg \lg n$ for any $a \geq 1$ and most of an inductive proof that $T(n) \geq bn \lg \lg n$ for any $b \leq 1$. Technically, we're still missing the base cases in both proofs, but we can be fairly confident at this point that $\boxed{\boldsymbol{T(n) = \Theta(n \log \log n)}}$.

1.6 References

Methods for solving recurrences by annihilators, domain transformations, and secondary recurrences are nicely described in George Lueker's paper "Some techniques for solving recurrences" (*ACM Computing Surveys* 12(4):419–436, 1980). The Master Theorem is presented in Chapter 4 of CLRS. Sections 1–3 and 5 of this handout were based on notes taken by Ari Trachtenberg of lectures by Ed Reingold; the notes were then revised by Jeff Erickson. Sections 4 and 6 by Erickson.