12 Solving Recurrence Relations

Recurrence relations are perhaps the most important tool in the analysis of algorithms. We have encountered several methods that can sometimes be used to solve such relations, such as guessing the solution and proving it by induction, or developing the relation into a sum for which we find a closed form expression. We now describe a new method to solve recurrence relations and use it to settle the remaining open question in the analysis of Fibonacci heaps.

Annihilation of sequences. Suppose we are given an infinite sequence of numbers, $A = \langle a_0, a_1, a_2, \ldots \rangle$. We can multiply with a constant, shift to the left and add another sequence:

$$\begin{aligned} kA &= \langle ka_0, ka_1, ka_2, \ldots \rangle, \\ \mathrm{L}A &= \langle a_1, a_2, a_3, \ldots \rangle, \\ A + B &= \langle a_0 + b_0, a_1 + b_1, a_2 + b_2, \ldots \rangle. \end{aligned}$$

As an example, consider the sequence of powers of two, $a_i = 2^i$. Multiplying with 2 and shifting to the left give the same result. Therefore,

$$\mathrm{L}A - 2A = \langle 0, 0, 0, \ldots \rangle.$$

We write $\mathrm{L}A - 2A = (\mathrm{L} - 2)A$ and think of $\mathrm{L} - 2$ as an operator that *annihilates* the sequence of powers of 2. In general, $\mathrm{L} - k$ annihilates any sequence of the form $\langle ck^i \rangle$. What does $\mathrm{L} - k$ do to other sequences $A = \langle c\ell^i \rangle$, when $\ell \neq k$?

$$\begin{aligned} (\mathrm{L} - k)A &= \langle c\ell, c\ell^2, c\ell^3, \ldots \rangle - \langle ck, ck\ell, ck\ell^2, \ldots \rangle \\ &= (\ell - k)\langle c, c\ell, c\ell^2, \ldots \rangle \\ &= (\ell - k)A. \end{aligned}$$

We see that the operator $\mathrm{L} - k$ annihilates only one type of sequence and multiplies other similar sequences by a constant.

Multiple operators. Instead of just one, we can apply several operators to a sequence. We may multiply with two constants, $k(\ell A) = (k\ell)A$, multiply and shift, $\mathrm{L}(kA) = k(\mathrm{L}A)$, and shift twice, $\mathrm{L}(\mathrm{L}A) = \mathrm{L}^2 A$. For example, $(\mathrm{L} - k)(\mathrm{L} - \ell)$ annihilates all sequences of the form $\langle ck^i + d\ell^i \rangle$, where we assume $k \neq \ell$. Indeed, $\mathrm{L} - k$ annihilates $\langle ck^i \rangle$ and leaves behind $\langle (\ell - k)d\ell^i \rangle$, which is annihilated by $\mathrm{L} - \ell$. Furthermore, $(\mathrm{L} - k)(\mathrm{L} - \ell)$ annihilates no other sequences. More generally, we have

FACT. $(\mathrm{L} - k_1)(\mathrm{L} - k_2) \ldots (\mathrm{L} - k_n)$ annihilates all sequences of the form $\langle c_1 k_1^i + c_2 k_2^i + \ldots + c_n k_n^i \rangle$.

What if $k = \ell$? To answer this question, we consider

$$\begin{aligned} (\mathrm{L} - k)^2 \langle ik^i \rangle &= (\mathrm{L} - k)\langle (i+1)k^{i+1} - ik^{i+1} \rangle \\ &= (\mathrm{L} - k)\langle k^{i+1} \rangle \\ &= \langle 0 \rangle. \end{aligned}$$

More generally, we have

FACT. $(\mathrm{L} - k)^n$ annihilates all sequences of the form $\langle p(i)k^i \rangle$, with $p(i)$ a polynomial of degree $n - 1$.

Since operators annihilate only certain types of sequences, we can determine the sequence if we know the annihilating operator. The general method works in five steps:

1. Write down the annihilator for the recurrence.
2. Factor the annihilator.
3. Determine what sequence each factor annihilates.
4. Put the sequences together.
5. Solve for the constants of the solution by using initial conditions.

Fibonacci numbers. We put the method to a test by considering the Fibonacci numbers defined recursively as follows:

$$\begin{aligned} F_0 &= 0, \\ F_1 &= 1, \\ F_j &= F_{j-1} + F_{j-2}, \quad \text{for } j \geq 2. \end{aligned}$$

Writing a few of the initial numbers, we get the sequence $\langle 0, 1, 1, 2, 3, 5, 8, \ldots \rangle$. We notice that $\mathrm{L}^2 - \mathrm{L} - 1$ annihilates the sequence because

$$\begin{aligned} (\mathrm{L}^2 - \mathrm{L} - 1)\langle F_j \rangle &= \mathrm{L}^2\langle F_j \rangle - \mathrm{L}\langle F_j \rangle - \langle F_j \rangle \\ &= \langle F_{j+2} \rangle - \langle F_{j+1} \rangle - \langle F_j \rangle \\ &= \langle 0 \rangle. \end{aligned}$$

If we factor the operator into its roots, we get

$$\mathrm{L}^2 - \mathrm{L} - 1 = (\mathrm{L} - \varphi)(\mathrm{L} - \overline{\varphi}),$$

where

$$\varphi = \frac{1 + \sqrt{5}}{2} = 1.618\ldots,$$

$$\overline{\varphi} = 1 - \varphi = \frac{1 - \sqrt{5}}{2} = -0.618\ldots.$$

The first root is known as the *golden ratio* because it represents the aspect ratio of a rectangular piece of paper from which we may remove a square to leave a smaller rectangular piece of the same ratio: $\varphi : 1 = 1 : \varphi - 1$. Thus we know that $(\mathrm{L} - \varphi)(\mathrm{L} - \overline{\varphi})$ annihilates $\langle F_j \rangle$ and this means that the j-th Fibonacci number is of the form $F_j = c\varphi^j + \overline{c}\,\overline{\varphi}^j$. We get the constant factors from the initial conditions:

$$\begin{aligned} F_0 &= 0 = c + \overline{c}, \\ F_1 &= 1 = c\varphi + \overline{c}\,\overline{\varphi}. \end{aligned}$$

Solving the two linear equations in two unknowns, we get $c = 1/\sqrt{5}$ and $\overline{c} = -1/\sqrt{5}$. This implies that

$$F_j = \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2} \right)^j - \frac{1}{\sqrt{5}} \left(\frac{1-\sqrt{5}}{2} \right)^j .$$

From this viewpoint, it seems surprising that F_j turns out to be an integer for all j. Note that $|\varphi| > 1$ and $|\overline{\varphi}| < 1$. It follows that for growing exponent j, φ^j goes to infinity and $\overline{\varphi}^j$ goes to zero. This implies that F_j is approximately $\varphi^j/\sqrt{5}$, and that this approximation becomes more and more accurate as j grows.

Maximum degree. Recall that $D(n)$ is the maximum possible degree of any one node in a Fibonacci heap of size n. We need two easy facts about the kind of trees that arise in Fibonacci heaps in order to show that $D(n)$ is at most logarithmic in n. Let ν be a node of degree j, and let $\mu_1, \mu_2, \ldots, \mu_j$ be its children ordered by the time they were linked to ν.

DEGREE LEMMA. The degree of μ_i is at least $i - 2$.

PROOF. Recall that nodes are linked only during the deletemin operation. Right before the linking happens, the two nodes are roots and have the same degree. It follows that the degree of μ_i was at least $i - 1$ at the time it was linked to ν. The degree of μ_i might have been even higher because it is possible that ν lost some of the older children after μ_i had been linked. After being linked, μ_i may have lost at most one of its children, for else it would have been cut. Its degree is therefore at least $i - 2$, as claimed. □

SIZE LEMMA. The number of descendents of ν (including ν) is at least F_{j+2}.

PROOF. Let s_j be the minimum number of descendents a node of degree j can have. We have $s_0 = 1$ and $s_1 = 2$. For larger j, we get s_j from s_{j-1} by adding the size of a minimum tree with root degree $j-2$, which is s_{j-2}. Hence $s_j = s_{j-1} + s_{j-2}$, which is the same recurrence relation that defines the Fibonacci numbers. The initial values are shifted two positions so we get $s_j = F_{j+2}$, as claimed. □

Consider a Fibonacci heap with n nodes and let ν be a node with maximum degree $D = D(n)$. The Size Lemma implies $n \geq F_{D+2}$. The Fibonacci number with index $D + 2$ is roughly $\varphi^{D+2}/\sqrt{5}$. Because $\overline{\varphi}^{D+2} < \sqrt{5}$, we have

$$n \geq \frac{1}{\sqrt{5}} \varphi^{D+2} - 1.$$

After rearranging the terms and taking the logarithm to the base φ, we get

$$D \leq \log_\varphi \sqrt{5}(n+1) - 2.$$

Recall that $\log_\varphi x = \log_2 x / \log_2 \varphi$ and use the calculator to verify that $\log_2 \varphi = 0.694\ldots > 0.5$ and $\log_\varphi \sqrt{5} = 1.672\ldots < 2$. Hence

$$\begin{aligned} D &\leq \frac{\log_2(n+1)}{\log_2 \varphi} + \log_\varphi \sqrt{5} - 2 \\ &< 2\log_2(n+1). \end{aligned}$$

Non-homogeneous terms. We now return to the annihilation method for solving recurrence relations and consider

$$a_j = a_{j-1} + a_{j-2} + 1.$$

This is similar to the recurrence that defines Fibonacci numbers and describes the minimum number of nodes in an *AVL tree*, also known as *height-balanced tree*. It is defined by the requirement that the height of the two subtrees of a node differ by at most 1. The smallest tree of height j thus consists of the root, a subtree of height $j - 1$ and another subtree of height $j - 2$. We refer to the terms involving a_i as the *homogeneous* terms of the relation and the others as the *non-homogeneous* terms. We know that $\mathrm{L}^2 - \mathrm{L} - 1$ annihilates the homogeneous part, $a_j = a_{j-1} + a_{j-2}$. If we apply it to the entire relation we get

$$\begin{aligned} (\mathrm{L}^2 - \mathrm{L} - 1)\langle a_j \rangle &= \langle a_{j+2} \rangle - \langle a_{j+1} \rangle - \langle a_j \rangle \\ &= \langle 1, 1, \ldots \rangle. \end{aligned}$$

The remaining sequence of 1s is annihilated by $\mathrm{L} - 1$. In other words, $(\mathrm{L} - \varphi)(\mathrm{L} - \overline{\varphi})(\mathrm{L} - 1)$ annihilates $\langle a_j \rangle$ implying that $a_j = c\varphi^j + \overline{c}\,\overline{\varphi}^j + c'1^j$. It remains to find

the constants, which we get from the boundary conditions $a_0 = 1$, $a_1 = 2$ and $a_2 = 4$:

$$\begin{array}{rcrcrcl} c & + & \overline{c} & + & c' & = & 1, \\ \varphi c & + & \overline{\varphi}\,\overline{c} & + & c' & = & 2, \\ \varphi^2 c & + & \overline{\varphi}^2\overline{c} & + & c' & = & 4. \end{array}$$

Noting that $\varphi^2 = \varphi + 1$, $\overline{\varphi}^2 = \overline{\varphi} + 1$, and $\varphi - \overline{\varphi} = \sqrt{5}$ we get $c = (5 + 2\sqrt{5})/5$, $\overline{c} = (5 - 2\sqrt{5})/5$, and $c' = -1$. The minimum number of nodes of a height-j AVL tree is therefore roughly the constant c times φ^j. Conversely, the maximum height of an AVL tree with $n = c\varphi^j$ nodes is roughly $j = \log_\varphi(n/c) = 1.440\ldots \cdot \log_2 n + \mathrm{O}(1)$. In words, the height-balancing condition implies logarithmic height.

Transformations. We extend the set of recurrences we can solve by employing transformations that produce relations amenable to the annihilation method. We demonstrate this by considering mergesort, which is another divide-and-conquer algorithm that can be used to sort a list of n items:

`Step 1.` Recursively sort the left half of the list.

`Step 2.` Recursively sort the right half of the list.

`Step 3.` Merge the two sorted lists by simultaneously scanning both from beginning to end.

The running time is described by the solution to the recurrence

$$\begin{array}{rcl} T(1) & = & 1, \\ T(n) & = & 2T(n/2) + n. \end{array}$$

We have no way to work with terms like $T(n/2)$ yet. However, we can transform the recurrence into a more manageable form. Defining $n = 2^i$ and $t_i = T(2^i)$ we get

$$\begin{array}{rcl} t_0 & = & 1, \\ t_i & = & 2t_{i-1} + 2^i. \end{array}$$

The homogeneous part is annihilated by $\mathrm{L} - 2$. Similarly, non-homogeneous part is annihilated by $\mathrm{L} - 2$. Hence, $(\mathrm{L} - 2)^2$ annihilates the entire relation and we get $t_i = (ci + \overline{c})2^i$. Expressed in the original notation we thus have $T(n) = (c \log_2 n + \overline{c})n = \mathrm{O}(n \log n)$. This result is of course no surprise and reconfirms what we learned earlier about sorting.

The Master Theorem. It is sometimes more convenient to look up the solution to a recurrence relation than playing with different techniques to see whether any one can make it to yield. Such a cookbook method for recurrence relations of the form

$$T(n) \quad = \quad aT(n/b) + f(n)$$

is provided by the following theorem. Here we assume that $a \geq 1$ and $b > 1$ are constants and that f is a well-behaved positive function.

MASTER THEOREM. Define $c = \log_b a$ and let ε be an arbitrarily small positive constant. Then

$$T(n) \quad = \quad \begin{cases} \mathrm{O}(n^c) & \text{if } f(n) = \mathrm{O}(n^{c-\varepsilon}), \\ \mathrm{O}(n^c \log n) & \text{if } f(n) = \mathrm{O}(n^c), \\ \mathrm{O}(f(n)) & \text{if } f(n) = \Omega(n^{c+\varepsilon}). \end{cases}$$

The last of the three cases also requires a usually satisfied technical condition, namely that $af(n/b) < \delta f(n)$ for some constant δ strictly less than 1. For example, this condition is satisfied in $T(n) = 2T(n/2) + n^2$ which implies $T(n) = \mathrm{O}(n^2)$.

As another example consider the relation $T(n) = 2T(n/2) + n$ that describes the running time of mergesort. We have $c = \log_2 2 = 1$ and $f(n) = n = \mathrm{O}(n^c)$. The middle case of the Master Theorem applies and we get $T(n) = \mathrm{O}(n \log n)$, as before.