4. Sequences that Count and Recurrence Relations

Sequences are useful for more than just scaring Calculus II students (if you have not yet encountered these objects in such a class, then go ask a friend who has and watch them shudder). The truth is, of course, that sequences can actually be used for lots of interesting tasks, including counting.

Defining the terms of a sequence to *count* something of interest is a popular "trick" used by many Computer Scientists and Mathematicians. And here's the best part: many times ***these counting-sequences feature recursive structure***! Indeed, the general "game" goes like this

Want to count something?	$\rightsquigarrow$	Define a sequence $\{a_n\}$ to do it!	$\rightsquigarrow$	Establish recursive structure and solve the recurrence equation

This is why many Discrete Math courses view the topic of "solving recurrence equations" as an advanced counting technique. However, solving recurrence equations is an interesting and useful skill Mathematicians use for all sorts of reasons (not simply as part of a counting strategy).

For this section we will review two examples of counting-sequences and then focus on solving recurrence equations.

4.1. Sequences that Count. Our first example will likely seem very familiar.

Example 4.1. *Suppose we want to count the number of non-repeating, length-n lists that are made using n distinct symbols. We can define*

$$a_n = \text{number of non-repeating } n\text{-lists using } n \text{ symbols}$$

There is only one length-0 list (the "empty list"), and so $a_0 = 1$. There is only one length-1 list (that uses one symbol), and so $a_1 = 1$, too. The number of length-2 lists made from two symbols can be computed using our multiplication principle, giving us $a_2 = 2$, and, similarly, we can compute that $a_3 = 6$.

However, there is recursive structure in the sequence $\{a_n\}$ that we can use, too. In particular every length-n list is a length-$(n-1)$ list with one extra symbol:

$$\text{length-}n \text{ list } = \underbrace{\_\,\_\,\_\,\cdots\,\_\,\_}_{n \text{ entries}} = \underbrace{\_\,\_\,\_\,\cdots\,\_}_{n-1 \text{ entries}}\,\_$$

There are n options for the final, n-th entry, depending on the $(n-1)$ symbols the first entries use. According to our multiplication principle, it follows that

$$a_n = (a_{n-1}) \cdot n = n \cdot a_{n-1}$$

When equipped with the initial condition $a_0 = 1$, recurrence equation can be used to define $n!$

Example 4.2. *Suppose we wish to count the number of length-n bit strings that do* not *contain a pair of consecutive* 0*'s. We first define the sequence*

$$a_n = \text{the number of length-}n\text{ bit strings not containing } 00$$

and explore some of its terms.

We can regard the case when $n = 0$ *as an "empty bit string," if we like, and note that this single string does not contain a pair of consecutive* 0*'s. Therefore* $a_0 = 1$*. Similarly, when* $n = 1$ *our bit strings have only one entry and so no consecutive* 0*'s are possible. Since there are two length-1 bit strings we have* $a_1 = 2$*.*

The following table summarizes the first few entries of our sequence.

n	bit strings	a_n
0	empty bit string	1
1	$\underline{0}$, $\underline{1}$	2
2	$\underline{0}\underline{1}$, $\underline{1}\underline{0}$, $\underline{1}\underline{1}$	3
3	$\underline{0}\underline{1}\underline{0}$, $\underline{0}\underline{1}\underline{1}$, $\underline{1}\underline{0}\underline{1}$ $\underline{1}\underline{1}\underline{0}$, $\underline{1}\underline{1}\underline{1}$	5
4	$\underline{0}\underline{1}\underline{0}\underline{1}$, $\underline{0}\underline{1}\underline{1}\underline{0}$, $\underline{0}\underline{1}\underline{1}\underline{1}$ $\underline{1}\underline{0}\underline{1}\underline{0}$, $\underline{1}\underline{0}\underline{1}\underline{1}$ $\underline{1}\underline{0}\underline{1}\underline{1}$ $\underline{1}\underline{1}\underline{1}\underline{0}$, $\underline{1}\underline{1}\underline{1}\underline{1}$	8

We're hoping you've noticed a familiar pattern emerging in the third column; in particular, the values of a_n *should remind of that very famous Fibonacci sequence!*

In particular, it appears as though our counting sequence a_n *satisfies the recurrence equation* $a_n = a_{n-1} + a_{n-2}$*, and, indeed, this is true. But how can we argue that this is true? (Give yourself a moment to see if you can come up with your own explanation.)*

One way to establish this recurrence equation is to note that every length-n bit-string is simply a length-$(n-1)$ *bit-string with one extra entry:*

$$\text{length-}n\text{ bit string } = \underbrace{\_\_\_\cdots\_\_}_{n\,\text{entries}} = \underbrace{\_\_\_\cdots\_}_{n-1\,\text{entries}} -$$

and that we can use a 1 *for the extra, n-th entry no matter what length-*$(n-1)$ *bit string precedes it (doing this will never create* 00*). However, we can use a* 0 *for the extra, n-th entry only if the previous,* $(n-1)$*-th entry is not also a* 0*; that is, the* $(n-1)$*-th entry must be a* 1*, and all other* $n-2$ *entries can be considered a length-*$(n-2)$ *bit string with no* 00*'s.*

Altogether, then, we are left with lists of the form

$$\underbrace{\_\_\_\cdots\_}_{n-1\,\text{entries}}\underline{1} \quad \textit{or} \quad \underbrace{\_\_\_\cdots\_}_{n-2\ \text{entries}}\ \underline{1}\underline{0}$$

It follows that the total number of allowed length-n bit strings equals the number of length-$(n-1)$ *strings added with the number of length-*$(n-2)$ *ones. In other words*

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

There are plenty of other sequences that are defined to "count things," many relating to topics in Computer Science (specifically to algorithms such as linear and binary searches) and to various branches of Mathematics. The two examples presented here may seem insufficient, perhaps, but if your understanding of them is thorough, then they really will provide an excellent foundation for using other counting-sequences, too (plus your instructor will be assigning some questions that will let you practice more examples, too).

4.2. Linear, Homogeneous Recurrence Equations. Recall that a recursively-defined sequence is given by a **recurrence equation** and **initial conditions**. For example, one might encounter the recursive sequences $\{a_n\}$ and $\{b_n\}$ given by

$$\begin{array}{ll} a_n = 5a_{n-1} & b_n = 2b_{n-1} - 4b_{n-2} \\ a_0 = 2 & b_1 = 2, \text{ and } a_0 = 3 \end{array}$$

We will also refer to recurrence equations like these as **recurrence relations**. Of course, recurrence equations and recurrence relations can be far more complicated than the lovely examples above.

Notice how for the examples $\{a_n\}$ and $\{b_n\}$ the functions relating "new" terms to "previous ones" only involve adding, subtracting and multiplying by constants? Those nice properties earn these recurrence equations a special name: these are examples of **linear recurrence equations**. There is one other property we should highlight, too, and that is that the recurrence relations above **do not feature a non-zero constant**. For instance, the sequence

$$c_n = 5c_{n-1} + 3$$

does have an extra constant, and it turns out that this makes it non-trivially more complicated to "solve" (although far from impossible).

Because the recursively defined sequences $\{a_n\}$ and $\{b_n\}$ have all of these nice properties – their recurrence equations are **linear** and they do not include a somewhat-problematic (non-zero) constant – we call their recurrence relations **linear, homogeneous recurrence equations**. (Those that feature constants, like $\{c_n\}$, are called **non-homogeneous**.)

Example 4.3. *Several recursively-defined sequences are provided below (although no initial conditions are provided). Make sure you can identify which ones are linear (but non-homogeneous), which ones are linear* and *homogeneous, and which ones are not linear.*

(1) $a_n = 2a_{n-1} + 3a_{n-2}$ *(linear, homogeneous)*

(2) $b_n = 2b_{n-1} + 3b_{n-2} + \pi$ *(linear, non-homogeneous)*

(3) $d_n = -3d_{n-1} \cdot d_{n-2}$ *(non-linear)*

(4) $e_n = 2n \cdot e_{n-1}$ *(non-linear)*

(5) $g_n = g_{n-1} - g_{n-2} + 4g_{n-3}$ *(linear, homogeneous)*

(6) $h_n = 2n \cdot h_{n-1} + 3n \cdot h_{n-2} - \sqrt{2}$ *(non-linear)*

Those readers familiar with Linear Algebra would do well to observe that **non-homogeneous, linear recurrence equations** express "upcoming" terms from a

sequence as **a linear combination** of previous ones. More generally, a **linear, homogeneous** recurrence equation is expressed as

$$a_n = f\,(\text{previous terms})$$

where the function f is only allowed to multiply previous terms against constants and then combine these expressions using addition and subtraction. To be repetitive but more mathematically precise, this means

$$\begin{aligned} a_n &= f\,(\text{previous terms}) \\ &= c_1\,a_{n-1} + c_2\,a_{n-2} + \cdots + c_k\,a_{n-k} \end{aligned}$$

where the coefficients c_i are constants.

Linear, homogeneous recurrence equations can be handled in relatively simple ways, even though there are a variety of them. Indeed, how "complex" they are depends on how many previous terms they use. That number k used in the expression up above? The larger k is, the more "previous terms" are used, and so we use its value to describe our recurrence relations. The **order** or **degree** of a linear, homogeneous recurrence equation is simply the number of "previous terms" on which it depends. The sequences $\{a_n\}$ and $\{b_n\}$ way at the beginning of this subsection are defined by linear, homogeneous recurrence equations of orders 1 and 2, respectively.

Example 4.4. *Make sure you can identify the order of each of the following linear, homogeneous recurrence equations.*

(1) $a_n = 2a_{n-1} + 3a_{n-2}$ *(order 2)*

(2) $g_n = g_{n-1} - g_{n-2} + 4g_{n-3}$ *(order 3)*

Our goal will be to "solve" linear, homogeneous recurrence equations of orders 1 or 2. Techniques for solving higher order ones exist, but we treat them as "beyond the scope of this course / textbook," meaning you'll have plenty of opportunities to examine order-3 and higher ones in other classes.

Before moving on with "solving" such equations, we offer one last comment: those readers who have worked onOrdinary Differential Equations will recognize the many adjectives presented here, namely "**linear** and **non-homogeneous**." These apply to differential equations in a way that is very similar to how they apply to our *recurrence equations* – even the solution techniques are similar!

4.3. Solving Recurrence Equations Using Iteration. Suppose we have a recursively-defined sequence, maybe one like

$$a_n = 3a_{n-1} \text{ with } a_0 = 4.$$

How might we go about finding the value of the term a_5? One option is to **iterate** the recurrence equation until we arrive at the initial condition, as follows:

$$a_5 = 3a_4 = 3 \cdot 3a_3 = 3 \cdot 3 \cdot 3a_2 = 3 \cdot 3 \cdot 3 \cdot 3a_1 = 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3a_0 = 3^5 \cdot 4$$

Take a moment to make certain that you understand how the values of, say, a_{10} and a_{200} can be discovered through **iteratoin**, too.

Example 4.5. *Consider the sequence $\{a_n\}$ that satisfies the linear, homogeneous recurrence relation $a_n = 3a_{n-1}$ with initial condition $a_0 = 4$. Iterating the recurrence equation allows us to conclude*

$$a_{10} = \underbrace{3a_9 = 3^2 a_8 = \cdots = 3^{10} \cdot 4}_{\text{total of 10 iterations}}$$

$$a_{200} = \underbrace{3a_{199} = 3^2 a_{198} = \cdots = 3^{200} \cdot 4}_{\text{total of 200 iterations}}$$

This iteration process can be used to determine the value of any term in the sequence. One merely looks for a pattern (which, if so desired, can be proven using Induction). For this example

$$a_n = \underbrace{\underbrace{3a_{n-1} = 3^2 a_{n-2} = \cdots = 3^k a_{n-k}}_{k \text{ iterations}} = \cdots = 3^{n-1} a_1 = 3^n a_0 = 3^n \cdot 4}_{n \text{ iterations}}$$

We then arrive at a formula for each term of the sequence, namely $a_n = 4 \cdot 3^n$.

Example 4.6. *Use the method of iteration to determine the value of a_5 for the sequence defined by*

$$a_n = 2a_{n-1} + a_{n-2} + 1 \text{ with } a_1 = 1, a_0 = 3.$$

Note that this is a linear, non-homogeneous recurrence equation; iteration will work for this all the same.

This "method of iteration" can be used on lots of recursively-defined sequences, but sometimes the recurrence equation is too complicated for a clear or simple pattern to emerge. (However, iterating the recurrence equation a few times is *always* a good idea when you're trying to understand a given sequence.) That's why you also want to learn about a more robust recipe for cooking up "solutions" to these problems, and that's precisely what we have in store.

Before doing this, though, let's make one last definition here, clarifying what it means to "**solve recurrence equation**." When we have a recursively defined sequence $\{a_n\}$, its recurrence equation is said to be **solved** when an explicit formula for each term, a_n, has been found – and this formula is not allowed to depend on previous terms (only on the index or input variable n). Such solutions are often called **closed form solution**.

In the example above we were given the recursive definition $a_n = 3a_{n-1}$ (with initial condition $a_0 = 4$) and, using the method of iteration, we discovered the **closed form solution** $a_n = 4 \cdot 3^n$.

4.4. Solving Recurrence Equations Using Algebra. An order-1, linear, homogeneous recurrence equation can always be solved iteratively (as we did with the

example above), and the closed form solutions have similar properties. In particular, they all have the form of **geometric sequences**, a fact we record as the following Theorem.

Theorem 5.1. *The linear, homogeneous, order-1 recurrence relation* $a_n = c_1 a_{n-1}$ *with initial condition* a_0 *has as the closed form solution*

$$a_n = a_0 (c_1)^n .$$

There is an analogous result for solutions to linear, homogeneous, order-2 recurrence equations, too.

Theorem 5.2. *The linear, homogeneous, order-2 recurrence relation*

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} \text{ with initial conditions } a_1 \text{ and } a_0$$

has as a closed form solution

$$a_n = \alpha\ (r_1)^n + \beta\, (r_2)^n$$
$$\text{or } a_n = \alpha\, r^n + \beta\, n\, r^n$$

where $\alpha, \beta \in \mathbb{R}$ *are determined by the initial conditions* a_0 *and* a_1*, and where* $r_1, r_2, r \in \mathbb{R}$ *are determined by* ***the charateristic equation****.*

The term "**characteristic equation**" is one we, obviously, need to explain – and we will! First, though, if you've taken Linear Algebra or Ordinary Differential Equations, then you will have heard of this phrase already (and the way we use it here is remarkably similar to how its used in those classes). Second, a *proof* of Theorem 5.2 will not be our focus, but interested readers can consult supplemental sections. Instead, we will emphasize how this Theorem can be used as a kind of "recipe" for producing solutions to these recurrence equations.

So what is this "**characteristic equation**" and how, exactly, can we use it to solve second order, linear, homogeneous recurrence equations? We'll focus on a particular example, say the following

$$a_n = 2a_{n-1} + 3a_{n-2}$$
$$\text{with } a_1 = 1, a_0 = 2$$

We can rewrite the recurrence relation as

$$a_n - 2a_{n-1} + 3a_{n-2} = 0$$

and then form the **characteristic equation**. We do this by creating a quadratic polynomial (often called the **characteristic polynomial**) by replacing a_n with the variable x^2, replacing a_{n-1} with x and replacing a_{n-2} with the constant 1:

$$\textbf{characteristic equation} : x^2 - 2x - 3 = 0$$

Old-fashioned algebra can be used here (either in the form of factoring, completing the square of applying the quadratic formula) to solve *this* equation. In this example we find two distinct (real number) roots:

$$r_1 = 3, r_2 = -1$$

We now know the basic template of the closed form solution:

$$a_n = \alpha\,(3)^n + \beta\,(-1)^n = \alpha 3^n + \beta\,(-1)^n$$

To finish the game, we need only determine the values of the constants α and β, and this is accomplished by using our initial conditions. For this problem we know that when $n = 0$ the term $a_0 = 2$ and that when $n = 1$ we obtain the term $a_1 = 1$. Plugging this into our equation above gives us two equations (with two unknowns):

$$2 = \alpha + \beta \qquad (n = 0)$$
$$1 = 3\alpha - \beta \qquad (n = 1)$$

Solving these equations tells us $\alpha = 3/4$ and $\beta = 5/4$. We now have our closed-form solution:

$$\boxed{a_n = \frac{3}{4}\,3^n + \frac{5}{4}\,(-1)^n}$$

This same technique will work even if we change the values of the coefficients 2 and 3 to arbitrary numbers like the ones we mentioned in the statement of the theorem, i.e. c_1 and c_2. There are two other situations we need to address, though: how do we proceed if there are not two distinct roots to the characteristic polynomial *and* how do we proceed if the characteristic polynomial only has complex roots? Altogether, then, there are three cases we would like to handle, and the following table summarizes two of them:

2nd Order, Linear, Homogeneous Recurrence Equation Outline

$$\boxed{a_n = c_1 a_{n-1} + c_2 a_{n-2} \text{ with } a_0, a_1}$$

(1) Form Characteristic Polynomial and Equation : $x^2 - c_1 x - c_2 = 0$

(2) Solve the characteristic equation

- (2a) Two Distinct Roots $r_1, r_2 \;\rightarrow a_n = \alpha\,(r_1)^n + \beta\,(r_2)^n$
- (2b) One Repeated Root $r \;\rightarrow a_n = \alpha\, r^n + \beta\, n\, r^n$

(3) Use the initial conditions to determine α, β

There is a way to handle a characteristic equation that produces complex roots (say one like $x^2 + 1 = 0$), but we will not discuss this case in this book. As a first pass through solving recurrence equations, we focus exclusively on the cases of **distinct, real roots** and **one repeated, real root**.

Example 4.7. *Solve the recurrence equation*

$$a_n = 10a_{n-1} - 25a_{n-2}$$

with initial conditions $a_0 = 3$ *and* $a_1 = 35$.

The characteristic equation can be formed and solved as follows:

$$x^2 - 10x + 25 = 0$$
$$(x-5)^2 = 0$$
$$x = 5$$

We have precisely one repeated root, i.e. $r = 5$. *Closed form solutions are then given by*

$$a_n = \alpha 5^n + \beta\, n\, 5^n$$

Plugging in $n = 0$ *and* $n = 1$, *respectively, yields the equations*

$$3 = \alpha$$
$$35 = 5\alpha + 5\beta$$

From this we find $\alpha = 3, \beta = 4$ *and*

$$a_n = 3 \cdot 5^n + 4n \cdot 5^n = (3 + 4n)5^n.$$

Example 4.8. *Find a closed form expression for the n-th Fibonacci number by solving the recurrence equation*

$$F_n = F_{n-1} + F_{n-2}$$

with initial conditions $F_0 = F_1 = 1$.