CS103A
Winter 2002

Handout 21
February 20, 2002

Recurrence Relations

Introduction

The number of bacteria in a colony doubles every hour. If a colony begins with five bacteria, how many will be present after 3 hours? After 5 hours? After 1900 hours? After n hours? Since the number of bacteria doubles every hour, the relationship $a_n = 2a_{n-1}$ holds whenever n is a positive integer. This relationship, together with the initial condition that $a_0 = 5$ uniquely determines a_n for all nonegative integers n. We can find a formula for a_n using only this information.

Many counting problems can be solved by finding relationships just like we did above. Such relationships are called **recurrence relations**, and are going to be the focus of the next few lectures. We're going to study a variety of counting problems that can be modeled using recurrence relations. We will develop methods here and in the next few handouts for finding explicit formulae for the terms of sequences that satisfy certain types of recurrence relations.

Recurrence Relations

Recall that a recursive definition of a sequence specifies one or more initial terms and a rule or two for determining subsequent terms for those that follow. Recursive definitions can be used to solve counting problems, and that can often be a good thing, because finding a closed formula for a recurrence relation and then using it to explicitly and quickly calculate a term for a particular integer is much quicker than calculating the term all the way up from the initial term—the **base case**, in a sense.

Handy Definition 1: A **recurrence relation** for a sequence of integers $\{a_n\}$ is a formula that expresses a_n in terms of one or more of the previous terms of a sequence—namely, the values $a_0, a_1, \ldots, a_{n-1}$, for all integers n with $n \geq n_0$ where n_0 is a nonnegative integer. A sequence is called a **solution** of a recurrence relation if its terms satisfy the recurrence relation.

Problem: Let $a_n = a_{n-1} - a_{n-2}$ for $n = 2, 3, 4, 5, \ldots$, and suppose that $a_0 = 3$ and $a_1 = 5$? What are a_4 and a_6?

This isn't too bad. We can first find a_2 using a_0 and a_1, and the find a_3 using a_1 and a_2. We can do all this like so:

$$\begin{aligned}
a_2 &= a_1 - a_0 = 5 - 3 = 2 \\
a_3 &= a_2 - a_1 = 2 - 5 = -3 \\
a_4 &= a_3 - a_2 = -3 - 2 = -5 \\
a_5 &= a_4 - a_3 = -5 - (-3) = -2 \\
a_6 &= a_5 - a_4 = -2 - (-5) = 3
\end{aligned}$$

Problem: Determine whether the sequence $\{a_n\}$ is a solution to the recurrence relation $a_n = 2a_{n-1} - a_{n-2}$ for $n = 2,3,4,5,\ldots$, where $a_n = 3n$ for every nonnegative integer n. Answer the same question where $a_n = 2^n$ and where $a_n = 5$.

Well, suppose that $a_n = 3n$ for every nonnegative integer n. Then, for $n \geq 2$ we have that $a_n = 2a_{n-1} - a_{n-2} = 2(3(n-1)) - 3(n-2) = 6n - 6 - 3n + 6 = 3n$. Therefore, $a_n = 3n$ sure seems to work. But what about this $a_n = 2^n$ idea? Plugging in the proposed solution, we see that $a_n = 2a_{n-1} - a_{n-2} = 2\ 2^{n-1} - 2^{n-2} = 4\ 2^{n-2} - 2^{n-2} = 3\ 2^{n-2} = 0.75\ 2^n$, which conflicts with the notion that $a_n = 2^n$. Therefore, $a_n = 2^n$ just doesn't work for us. Nope, no-sirree Bob! Finally, $a_n = 5$—how about it? Well, suppose that $a_n = 5$. Then for every integer n greater than or equal to 2, we should have that $a_n = 2a_{n-1} - a_{n-2} = 2\ 5 - 5 = 10 - 5 = 5$.

The initial conditions for a sequence specify the terms that precede the first term where the recurrence relation takes effect. For instance, in the first example, $a_0 = 3, a_1 = 5$ are the initial conditions. The recurrence relation and initial conditions uniquely determine a sequence. This is clearly the case, since a recurrence relation, together with the initial conditions, provide a recursive definition of the sequence. Any term of the sequence can be found from the initial conditions using the recurrence relation a sufficient number of times. However, there are better ways for computing the terms of certain classes of sequences defined by recurrence relations and the initial conditions. We'll discuss them soon enough, my little angels. Just wait.

Modeling with Recurrence Relations

We can use recurrence relations to model a wide variety of problems, such as finding compound interest, counting the number of ways place dominoes on an $n \times 2$ board, determining the number of moves in the Tower of Hanoi puzzle, and counting bit strings with certain properties.

Problem: Suppose that a person deposits ten thousand bucks in a savings account at a bank yielding 11% per year with interest compounded annually. How much will be in the account after 30 years?

To solve this problem, just let P_n denote the amount of cash in the account after n years. Since the amount in the account after n years equals the amount in the account after n-1 years plus the interest for the nth year, we see that the sequence $\{P_n\}$ satisfies the recurrence relation:

$$P_n = P_{n-1} + 0.11P_{n-1} = 1.11P_{n-1}$$

where the initial condition is simply stated as $P_0 = 10{,}000$.

We are totally entitled to use an iterative approach to find a closed formula for P_n. Note that:

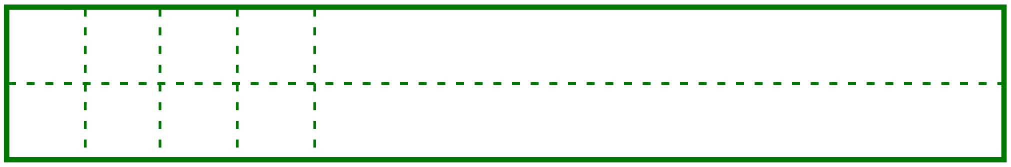
$P_0 = P_0$
$P_1 = 1.11P_0$
$P_2 = 1.11P_0 = (1.11)^2 P_0$
$P_3 = 1.11P_2 = (1.11)^3 P_0$
....
$P_n = 1.11P_{n-1} = (1.11)^n P_0$

When we insert the initial condition $P_0 = 10{,}000$, the formula $P_n = 1.11P_{n-1} = (1.11)^n 10{,}000$ is obtained. We can use mathematical induction to establish its validity. That the formula is valid for $n = 0$ is a consequence of the initial condition. Now, assume that $P_n = (1.11)^n 10{,}000$. Then, from the recurrence relation and the inductive hypothesis, $P_{n+1} = (1.11)P_n = (1.11)(1.11)^n 10{,}000 = (1.11)^{n+1} 10{,}000$. This shows that the explicit formula for P_n is valid. Sweet.

Substituting $n = 30$ into the formula $P_n = (1.11)^n 10{,}000$ shows that after 30 years the account contains $P_{30} = \$228{,}922.97$. If only banks offered that much interest on savings accounts, we'd all be set.

Problem: How many ways are there to tile a $2 \times n$ strip with 2×1 dominoes? Actually, let's just worry about coming up with a recurrence relation and listing the first few values of the related sequence. Deep breath.

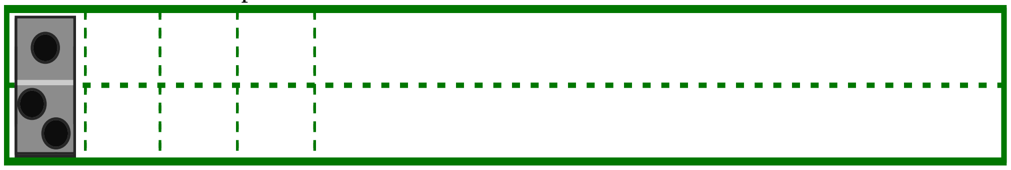
This may seem complicated. After all, it seems that dominoes can be placed horizontally or vertically, but that a horizontally placed domino mandates that a second horizontally placed domino be placed directly below or above it. It seems like the number of possibilities grows tentacles much more quickly than we can comprehend. Not to worry. Let's go nuts artistically and draw out $2 \times n$ strip.

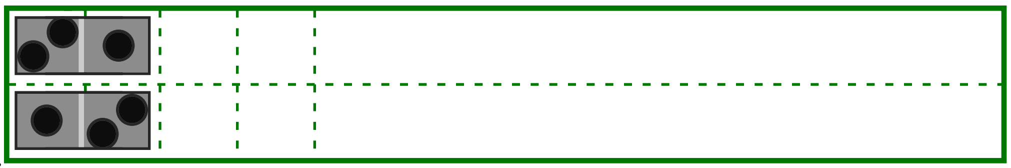


Okay, so it doesn't look as dazzling in the photocopy, but trust me. it's beautiful.

Aaaaaanway, let's not think so big, rather than thinking about the number of ways to cover the entire strip, let's just think about the number of ways to cover just the left end of the strip—that is the two smallest dashed boxes at the left of the strip above.

Well, we could cover both using a single domino, as in the drawing to the right. Or we could use two dominoes, each placed horizontally to cover one of the two squares that need to be covered.





Well, it's east to convince ourselves that these are the only two ways to cover the leftmost column of squares of our strip. But it tells us a great deal! It tells us that the number of ways to cover an $n \times 2$ strip is equal to the number of ways to cover an $(n-1) \times 2$ strip plus the number of ways to cover an $(n-2) \times 2$ strip.

The number of tilings can be modeled using a recurrence relation, which is nice, cause that's what today's topic is. In particular, there is exactly one way to tile a 1×2 strip using 1×2 dominoes, and there are precisely two ways to tile a 2×2 strip using 1×2 dominoes. Hmmmm… let's summarize:

$$T_1 = 1$$
$$T_2 = 2$$

Now what about more complicated boards? Well, we should translate our observation from above about how to tile an $n \times 2$ strip:

$$T_n = T_{n-1} + T_{n-2}$$

Coincidentally, this recurrence relation is exactly the same as that used to calculate the Fibonacci numbers.

One thing two note about this particular recurrence is that the nth term is defined in terms of two earlier values of the sequence. This feature of the recurrence makes it more difficult to solve that that for the bank account example. However, it is still solvable, and we'll see an example soon enough where we get to actually compute a closed formula, which solves the recurrence for the tiling problem. I can't wait.

An Example

Problem: Define a **ternary string** as a string which contains only 1's, 2's, and 3's. Construct a recurrence relation to represent the number ternary strings of length n free of double digits. Provide a recurrence relation which models the counting problem, and then solve it.

The easiest way to construct the recurrence relation for this one was to realize that, in promoting a legal ternary string from length n-1 to another legal string of length n, we can append either of two characters to the end. Specifically, if a legal string of length n - 1 ends in a 0, then we can append either a 1 or a 2 to the end to get a legal string of length n. This leads to the simple formula:

$$T_1 = 3$$
$$T_n = 2T_{n-1}$$

I specifically omitted the n = 0 case, because it interferes with the recurrence relation a tad—I'll show how to come up a solution for this. Anyway, it's pretty clear that this recurrence has the solution:

$$T_n = 3\ \ 2^{n-1}, n \geq 1$$

Unfortunately, this doesn't really work for n = 0, since the empty string is a legal ternary string in this case, but $T_0 = 3\ \ 2^{-1} = 3/2$, and we all know that $3/2 \neq 1$.

One way to fix this puppy is to let $[n = 0]$ be an expression which evaluates to 1 when n = 0 and to 0 otherwise. Then we can just rewrite the formula above to embrace the n = 0 case as well.

Ah, romance.

$$T_n = 3\ 2^{n-1} - [n = 0]/2$$

Problem: Find a recurrence relation and give initial conditions for the number of bit strings of length n that do not contain two consecutive 0s. How many such bit strings are there of length five?

Well, let B_n be the number of bit strings of length n that do not have two consecutive zeros. To obtain a recurrence relation for $\{B_n\}$, note that the number of bit strings of length n that do not have two consecutive 0s equal the number of such bit strings ending in 0 plus the number of bit strings ending in 1. We can assume that n is greater than or equal to three, so that only bit strings of length three or more need be considered.

The valid bit strings of length n ending with 1 are precisely the bits strings of length n - 1 with no consecutive zeros with a 1 appended onto the end. Consequently, there are B_{n-1} such bit strings. Valid bit strings of length n ending in a zero must have 1 as their second to last bit; otherwise, they would end with a pair of 0s, and that just can't be. It follows that the legal bit strings of length n ending in zero are the valid bit strings of length n - 2 with 10 tacked on to the end. Consequently, there are B_{n-2} such bit strings.

We conclude that, for n greater than or equal to 3, that:

$$B_n = B_{n-1} + B_{n-2}$$

The initial conditions are $B_1 = 2$, since both bit strings of length one, 0 and 1, do not have two consecutive 0s, and $B_2 = 3$. To obtain B_5, we simply use the recurrence relation three more times to find that:

$$B_3 = B_2 + B_1 = 3 + 2 = 5$$
$$B_4 = B_3 + B_2 = 5 + 3 = 8$$
$$B_5 = B_3 + B_2 = 8 + 5 = 13$$

Problem: A computer system considers a string of decimal digits to be a valid code word if and only if it contains an even number of zero digits. For instance, 1230407869 is valid, whereas 3141529046 is not. Let V_n be the number of valid n-digit code words; find a recurrence for V_n.

I think the base case is the easiest to see first—$V_1 = 9$, since there are 10 one-digit strings and only one of them is illegal. A recurrence relation can be derived for this sequence by considering how a valid n-digit string can be obtained from strings of n - 1 digits.

First, a valid string of n digits can be obtained by appending a nonzero digit to the end of a valid codeword of length n-1. This appending, then, can be done in nine different ways, so that a valid string of length n can be formed in this particular manner in $9V_{n-1}$ ways.

Second, a valid string of n digits can be obtained by appending a 0 to the end of a length-(n-1) codeword that is not valid. The number of ways this can be done equals the number of invalid $(n-1)$-digit strings. Since there are $10^{n-1} - V_{n-1}$ illegal code words of length n-1, $10^{n-1} - V_{n-1}$ valid code words can be formed by simply appending a zero to the end of each.

Since all valid strings of length n are produced in one of two ways, it follows that there are

$$\begin{aligned} V_n &= (9V_{n-1}) + (10^{n-1} - V_{n-1}) \\ &= 8V_{n-1} + 10^{n-1} \end{aligned}$$

valid code words of length n.

Problem: Let's look at the Towers of Hanoi problem. A popular puzzle of the late 19th century, the Tower of Hanoi, consists of three pegs mounted on a board together with disks of different sizes. Initially these disks are placed on the first peg, with the largest on the bottom. The rules of the puzzle allow disks to be moved one at a time from one peg to another as long as a disk is never placed on top of a smaller one. The goal is to move the entire tower from the original peg to a second one, but obeying the rules with every move.

If we let H_n denote the number of moves needed to solve the Tower of Hanoi problem with n disks, then we can set up a recurrence relation for the sequence $\{H_n\}$ as follows.

Begin with n disks on one peg. We can transfer the top n - 1 disks, following the rules of the puzzle, to peg 3 using H_{n-1} moves, keeping the largest disk fixed during all of these moves. Then, we use one move to transfer the largest disk to the second peg. We can transfer the n-1 disks from peg 3 to peg 2 using an additional H_{n-1} moves, placing them on top of the largest disk, which always stays fixed on the bottom of peg 2. Moreover, it is easy to see that the puzzle cannot be solved using fewer steps, so that our recurrence formula is simply given as:

$$H_n = 2H_{n-1} + 1$$

The initial condition is $H_1 = 1$.

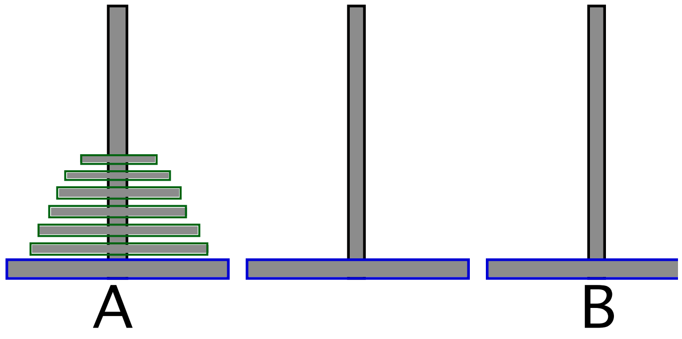
We can use the iterative approach that we used for the bank example to solve this particular recurrence relation. Note that $H_1 = 1$, since a tower of height 1 can be transferred from peg 1 to peg 2, according to the rules of the puzzle, in one move. Look what happens; it's lovely.

$$
\begin{aligned}
H_n &= 2H_{n-1} + 1 \\
&= 2\left(2H_{n-2} + 1\right) + 1 = 2^2 H_{n-2} + 2 + 1 \\
&= 2^2\left(2H_{n-3} + 1\right) + 2 + 1 = 2^3 H_{n-3} + 2^2 + 2 + 1 \\
&\ \ldots\ldots \\
&= 2^{n-1} H_1 + 2^{n-2} + 2^{n-3} + 2^{n-4} + \cdots + 2^2 + 2^1 + 1 \\
&= 2^{n-1} + 2^{n-2} + 2^{n-3} + 2^{n-4} + \cdots + 2^2 + 2^1 + 1 \\
&= \sum_{0 \le i \le n-1} 2^i = \frac{2^n - 1}{2 - 1} = 2^n - 1
\end{aligned}
$$

Note that I've used the recurrence relation repeatedly to express H_n in terms of previous values of the sequence. In the next to last equality, the initial condition that $H_1 = 1$ was used, and the last equality is based on the formula for the sum of the terms of a geometric series.

An Example

Problem: Find the shortest sequence of moves that transfers a tower of n disks from the left peg A to the right peg B, if direct moves between A and B are disallowed. (Each move must be to or from the middle peg. As usual, a larger disk must never appear above a smaller one.)



This isn't all that different than the normal Tower of Hanoi problem. If we let T_n denote the number of disk moves required to move a tower of height n in this manner. We first need to move the top n - 1 disks from A to B (requiring T_{n-1} disk moves,) then move the largest disk from A to the center peg. Next we should transfer the n - 1 disks from B back to A (requiring an additional T_{n-1} moves), and move the largest disk to peg B. Then we need only transfer the tower of height n - 1 from A to B again, requiring T_{n-1} moves. Therefore, the recurrence relation is given as:

$$
\begin{aligned}
T_n &= 0 \qquad && n = 0 \\
T_n &= 3T_{n-1} + 2 && n > 0
\end{aligned}
$$

We can solve this baby by repeated substitution, in exactly the same way we solved the original Tower of Hanoi recurrence.

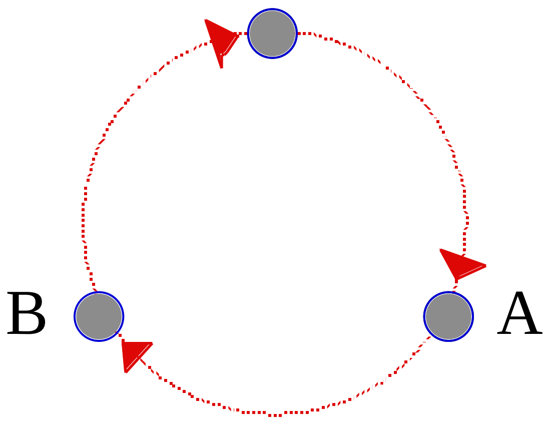
$$\begin{aligned}T_n &= 3T_{n-1} + 2\\ &= 3(3T_{n-2} + 2) + 2 = 3^2 T_{n-2} + 3\ 2 + 2\\ &= 3^2(3T_{n-3} + 2) + 3\ 2 + 2 = 3^3 T_{n-3} + 3^2\ 2 + 3\ 2 + 2\\ &= 3^3(3T_{n-4} + 2) + 2\sum_{0\le j<3} 3^j = 3^4 T_{n-4} + 2\sum_{0\le j<4} 3^j\\ &\ldots\ldots\\ &= 3^k T_{n-k} + 2\sum_{0\le j<k} 3^j\end{aligned}$$

This is also true when $k = n$, so that

$$\begin{aligned}T_n &= 3^n T_0 + 2\sum_{0\le j<n} 3^j\\ &= 0 + 2\frac{3^n - 1}{3-1} = 3^n - 1\end{aligned}$$

gives the required number of moves to transfer a tower of height n from peg A to peg B under the specified constraints.

Problem: Let Q_n be the minimum number of moves needed to transfer a tower of n disks from A to B, if all moves must be clockwise—that is, from A to B, or from B to the other peg, or from the other peg to A. Also, let R_n be the minimum number of moves needed to go from B back to A under this restriction. Justify why the following coupled set of recurrences applies:



$$Q_n = \begin{cases} 0; & \text{if } n = 0\\ 2R_{n-1} + 1 & \text{if } n > 0\end{cases} \qquad R_n = \begin{cases} 0; & \text{if } n = 0\\ Q_n + Q_{n-1} + 1 & \text{if } n > 0\end{cases}$$

I want to emphasize the clockwise constraint of the problem; rather than drawing the three pegs side by side in a line, I place them equidistantly on a circle, the A peg at 4 o'clock, the B peg at 8 o'clock, and the third peg at high noon. Also note: moving a tower of height n from peg A to B is the same as moving a tower from B to the other peg or from the other peg to A. By definition, each requires Q_n individual disk moves. Similarly, moving a tower of height n from B to A is identical to moving any tower of height n from A to the other peg of from the other peg to B, and by definition that requires R_n moves. Note that R_n isn't necessarily equal to $2Q_n$—moving a tower of height n from B to A doesn't necessarily require that the tower be fully and intermittently transferred to the other peg.

Anyway, to move the tower of height n from A to B, we need to move all but the largest disk to the other peg so that the largest disk can be transferred from A directly to B. Only after we've placed the largest disk where it needs to be should we transfer the tower of height n - 1 from the other peg to B. Moving a tower of height n from A to the other peg requires, by definition, R_{n-1} disk moves. Then we require a single disk move to move the largest disk, and then an additional R_{n-1} disk moves to finish off the job. That justifies the following:

$$\begin{aligned} Q_n &= 0 && n = 0 \\ Q_n &= 2R_{n-1} + 1 && n > 0 \end{aligned}$$

Now, trying to move a tower of height n from B back to A clearly requires that the largest disk be moved twice—once from B to the other peg, and once more from the other peg to A. That means that the tower of height n - 1 has to be moved out of the way two times, and then moved a third time to place it on the destination peg. We must make R_{n-1} disk moves to move the top n-1 disks to peg A, so that the largest disk can be moved to the other peg. Then we must make Q_{n-1} moves to transfer the (n-1)-tower from B back to A, so that the largest disk can be moved from the other peg to A. Finally, we can make R_{n-1} disk moves to move the (n-1)-tower from B to A. The total number of disk moves is then given as:

$$\begin{aligned} R_n &= 0 && n = 0 \\ R_n &= R_{n-1} + 1 + Q_{n-1} + 1 + R_{n-1} && n > 0 \end{aligned}$$

But we know $Q_n = 2R_{n-1} + 1$ from the first half of the solution, so that:

$$\begin{aligned} R_n &= 0 && n = 0 \\ R_n &= Q_n + Q_{n-1} + 1 && n > 0 \end{aligned}$$

One of the first recurrence relation problems I covered above answers the question: How many ways are there to tile an $n \times 2$ board using 2×1 dominoes? Our discussion brought us to the following recurrence relation:

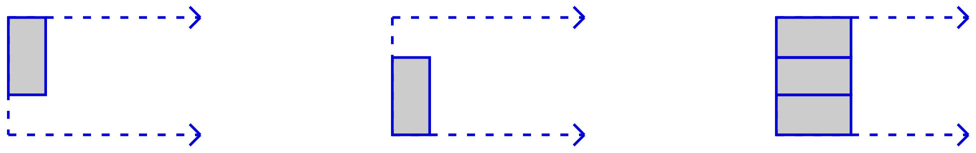
$$\begin{aligned} T_n &= 1 && n = 0,1 \\ T_n &= T_{n-1} + T_{n-2} && n \geq 2 \end{aligned}$$

An Example

Problem: Now consider a similar but more difficult question: How many ways are there to tile an $n \times 3$ board using 2×1 dominoes? Model this counting problem using a coupled set of recurrence relations which in theory could be solved to give you a closed formula.

One thing to note is that whatever the solution is, it should evaluate to zero for odd n. But I don't even want you to solve the set of recurrence relations—I just want the recurrence relation.

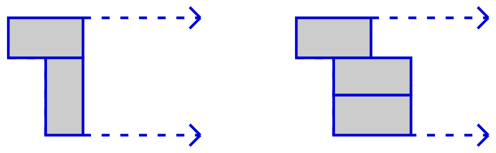
Whenever you are constructing a recurrence relation, you always try to break down the problem into smaller similar sub-problems. In this case, we should realize that every $3 \times n$ board must be partially tiled in one of three ways!



One important thing to note is that the first two cases are distinct but computationally identical. If we let U_n denote the number of ways to tile a $3 \times n$ board, then clearly U_n is related to the number of ways you can tile a U_{n-2} board. Check out the first two cases—the exposed portion is almost related to U_{n-1}, except we have that annoying lip jutting out to the left. Well, that annoying lip makes **that** tiling problem a completely different tiling problem—one that needs a new name. I'm feeling alphabetical, so let's let V_n represent the number of ways to tile a $3 \times n$ board with a big fat upper lip. Then all of a sudden, we can come up with a recurrence relation for U_n (recalling that the first and second cases are computationally identical and have no single tiling in common.)

$$U_n = U_{n-2} + 2V_{n-1}$$

Now, we wonder: Can V_n be related to smaller versions of U and V? Let's look at a generic V_n board, and see how it can be tiled:



That means that V_n can be computed as follows:

$$V_n = U_{n-1} + V_{n-2}$$

That gives us the entire system of equations, provided we seed them with some initial conditions:

$$U_n = \begin{cases} 1 & n = 0 \\ 0 & n = 1 \\ U_{n-2} + 2V_{n-1} & n \geq 2 \end{cases} \quad V_n = \begin{cases} 0 & n = 0 \\ 1 & n = 1 \\ U_{n-1} + V_{n-2} & n \geq 2 \end{cases}$$

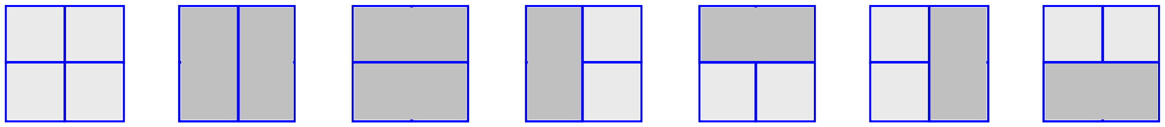
This is a pretty tough one. Very cool, but difficult.

An Example

Problem: How many ways are there to build a $2 \times 2 \times n$ pillar using $1 \times 1 \times 2$ bricks? Model this counting problem using another coupled set of recurrence relations that can in theory be solved to give you a closed formula. Make sure you understand the $n \times 3$ domino board problem before tackling this one, since your target set of recurrences will be similar in structure to that given there.

Let S_n represent the number of ways to build a $2 \times 2 \times n$ pillar using $2 \times 1 \times 1$ bricks, and let T_n be the number of ways to build such a pillar with a flat $2 \times 1 \times 1$ notch missing.

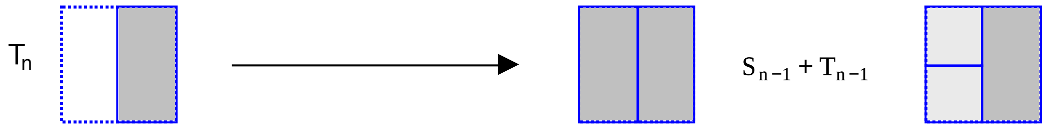
Whenever you are constructing a recurrence relation, you always try to break down the problem into smaller yet similar subproblems. In this case, we should realize that every $2 \times 2 \times n$ pillar must be partially built in one of seven ways!



The first of the seven corresponds to the placement of four bricks, each of which is standing on its 1×1 side, to create a $2 \times 2 \times 2$ foundation for the S_{n-2} pillars of height $n-2$ that can be built on top of it. The next two illustrate the two ways to build a $2 \times 2 \times 1$ foundation for the S_{n-1} $2 \times 2 \times (n-1)$ pillars that can used to complete the full construction. Note that the second and third partials are distinct but computationally identical cases, so both cases contribute the same number to the final answer. Foundations 4 through 7 are each the result of placing two tall-standing bricks aside a flat one. Note that each of these four is structurally different from the other three, and that all four are different from foundations 1, 2, and 3. In order to build a full $2 \times 2 \times n$ pillar from each of these four "$2 \times 2 \times 1.5$", we need to choose one of the T_{n-1} ways to build a $2 \times 2 \times (n-1.5)$ skyscraper and rest it on top of it. These seven cases form a partition of all the ways one can start to build a pillar of height n, and based on that observation, we can state with a good amount of confidence that the recurrence relation for S_n is given by:

$$S_n = \begin{cases} 1 & n = 0 \\ 2 & n = 1 \\ 2S_{n-1} + S_{n-2} + 4T_{n-1} & n \geq 2 \end{cases}$$

Formulating the recurrence relation for T_n is much simpler, because there are only two distinct ways to begin building.



$$S_n = \begin{cases} 1 & n = 0 \\ 2 & n = 1 \\ 2S_{n-1} + S_{n-2} + 4T_{n-1} & n \geq 2 \end{cases} \qquad T_n = \begin{cases} 0 & n = 0 \\ 1 & n = 1 \\ S_{n-1} + T_{n-1} & n \geq 2 \end{cases}$$

An Example

Problem: A space probe has discovered that organic material on Mars has DNA composed of five symbols, denoted by $\{a,b,c,d,e\}$, instead of the four components in earthling DNA. The four pairs cd, ce, ed, and ee never occur consecutively in a string of Martian DNA, but any string without forbidden pairs is possible. (Thus, bbcda is forbidden but bbdca is just fine.) We're curious how many Martian DNA strings of length n are possible? (When n = 2, the answer is 21, because left and right ends of a string are distinguishable.) Let a_n stand for the number of legitimate DNA strands that end in either c or e, and let b_n stand for those that don't. Arrive at a coupled recurrence relation for each of the a_n and the b_n. Understand that the real answer we're interested in is $a_n + b_n$.

Consider any of the a_{n-1} strands ending in c or e.
Now consider adding each of the five bases:

- a: forms one of the b_n DNA strands
- b: forms one of the b_n DNA strands
- c: forms one of the a_n DNA strands (but note that it ends in c, hence a_n and not b_n)
- d: forms an illegal strand.
- e: bzzzzt! forms another illegal strand

Consider any of the legal b_{n-1} strands ending in a, b, or d, and then consider what happens when you tack on one more base:

- a: forms one of the b_n DNA strands
- b: forms one of the b_n DNA strands
- c: forms one of the a_n DNA strands (and once again note it's a_n and not b_n)
- d: forms one of the b_n DNA strands
- e: forms one of the a_n DNA strands (for the same reason)

The recurrence relations that follow here are:

$$a_n = \begin{cases} 0 & n = 0 \\ 2 & n = 1 \\ a_{n-1} + 2b_{n-1} & n \geq 2 \end{cases} \qquad b_n = \begin{cases} 1 & n = 0 \\ 3 & n = 1 \\ 2a_{n-1} + 3b_{n-1} & n \geq 2 \end{cases}$$