there’s just one way to tile a 2 x 0 rectangle with dominoes, namely to use no dominoes; therefore \( T_0 = 1 \). (This spoils the simple pattern \( T_n = n \) that holds when \( n = 1, 2, \) and 3; but that pattern was probably doomed anyway, since \( T_0 \) wants to be 1 according to the logic of the situation.) A proper understanding of the null case turns out to be useful whenever we want to solve an enumeration problem.

Let’s look at one more small case, \( n = 4 \). There are two possibilities for tiling the left edge of the rectangle—we put either a vertical domino or two horizontal dominoes there. If we choose a vertical one, the partial solution is \( \square \) and the remaining 2 x 3 rectangle can be covered in \( T_3 \) ways. If we choose two horizontals, the partial solution \( \square \) can be completed in \( T_1 \) ways. Thus \( T_4 = T_3 + T_1 = 5 \). (The five tilings are \( UI, IR, EI, EII, \) and \( EIII \).)

We now know the first five values of \( T_n \):

<table>
<thead>
<tr>
<th>( n )</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T_n )</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>5</td>
</tr>
</tbody>
</table>

These look suspiciously like the Fibonacci numbers, and it’s not hard to see why: The reasoning we used to establish \( T_4 = T_3 + T_2 \) easily generalizes to \( T_n = T_{n-1} + T_{n-2} \), for \( n \geq 2 \). Thus we have the same recurrence here as for the Fibonacci numbers, except that the initial values \( T_0 = 1 \) and \( T_1 = 1 \) are a little different. But these initial values are the consecutive Fibonacci numbers \( F_1 \) and \( F_2 \), so the \( T \)'s are just Fibonacci numbers shifted up one place:

\[
T_n = F_{n+1}, \quad \text{for } n \geq 0.
\]

(We consider this to be a closed form for \( T_n \), because the Fibonacci numbers are important enough to be considered “known!” Also, \( F_n \) itself has a closed form (6.123) in terms of algebraic operations.) Notice that this equation confirms the wisdom of setting \( T_0 = 1 \).

But what does all this have to do with generating functions? Well, we’re about to get to that—there’s another way to figure out what \( T_n \) is. This new way is based on a bold idea. Let’s consider the “sum” of all possible 2 x \( n \) tilings, for all \( n \geq 0 \), and call it \( T \):

\[
T = 1 + \square + \square + \square + \square + \square + \cdots. \quad (7.1)
\]

(The first term ‘1’ on the right stands for the null tiling of a 2 x 0 rectangle.) This sum \( T \) represents lots of information. It’s useful because it lets us prove things about \( T \) as a whole rather than forcing us to prove them (by induction) about its individual terms.

The terms of this sum stand for tilings, which are combinatorial objects. We won’t be fussy about what’s considered legal when infinitely many tilings