

We immediately see that $n = 1$ is an easy case: just move the single disk from *source* to *dest*.

In the harder case, $n > 1$, we are forced to use the remaining pole (other than *source* and *dest*); let us call it *spare*. If $n = 2$, for example, we can move the smaller disk from *source* to *spare*, then move the larger disk from *source* to *dest*, and then finally move the smaller disk from *spare* to *dest*. (See Figure 2.19.)

This gives us a clue to solving the general case, $n > 1$. Assume for the moment that we have an auxiliary algorithm to move a tower of $(n-1)$ disks from one pole to another. Thus we can proceed as follows: move a tower of $(n-1)$ disks from *source* to *spare* (using the auxiliary algorithm); then move a single disk from *source* to *dest*; then move a tower of $(n-1)$ disks from *spare* to *dest* (again using the auxiliary algorithm). (See Figure 2.20.)

But of course we do not need an auxiliary algorithm to move a tower of $(n-1)$ disks: we just use the same algorithm recursively! Thus we have derived Algorithm 2.18.

To estimate how long the monks of Hanoi will take to move the tower of 64 disks, we need to analyze the algorithm. The characteristic operations are of course single-disk moves. Let $moves(n)$ be the number of single-disk moves required to move a tower of n disks from one pole to another. Then we can immediately write down the following equations:

$$moves(n) = 1 \quad \text{if } n = 1 \quad (2.8a)$$

$$moves(n) = 1 + 2 \, moves(n - 1) \quad \text{if } n > 1 \quad (2.8b)$$

Again we have a pair of recurrence equations. Again we skip the derivation and simply state the solution:

$$moves(n) = 2^n - 1 \quad (2.9)$$

Thus, for example, $moves(1) = 1$, which is obviously correct, and $moves(2) = 3$, which we already know from Figure 2.19. If you are mathematically inclined, you should verify the solution (2.9) by induction.

The Towers of Hanoi algorithm has time complexity $O(2^n)$. In fact, it is the first $O(2^n)$ algorithm we have encountered.

To move a tower of n disks from *source* to *dest* (where n is a positive integer):

1. If $n = 1$:
 - 1.1. Move a single disk from *source* to *dest*.
2. If $n > 1$:
 - 2.1. Let *spare* be the remaining pole, other than *source* and *dest*.
 - 2.2. Move a tower of $(n-1)$ disks from *source* to *spare*.
 - 2.3. Move a single disk from *source* to *dest*.
 - 2.4. Move a tower of $(n-1)$ disks from *spare* to *dest*.
3. Terminate.

Algorithm 2.18 Recursive algorithm for Towers of Hanoi.