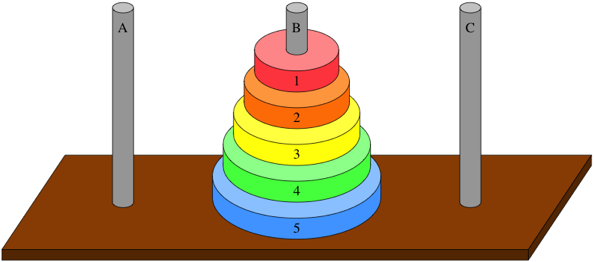
**<Towers of Hanoi(하노이의 탑)>**

The goal is to move all n disks from peg A to peg B:



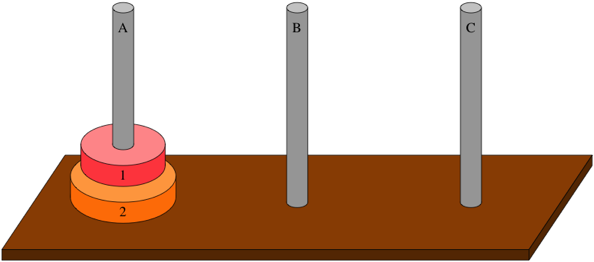


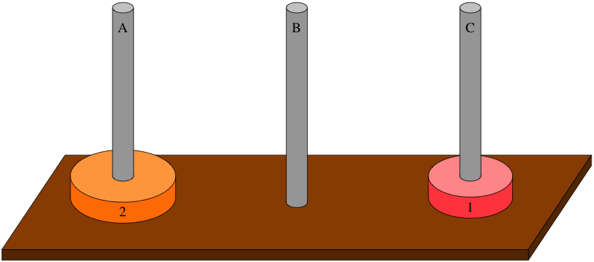
Sounds easy, right? It's not quite so simple, because you have to obey two rules:

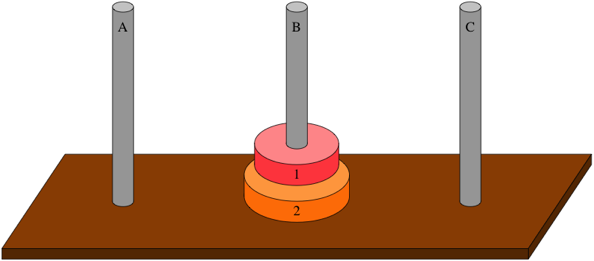
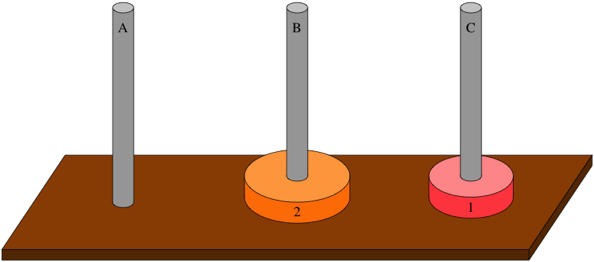
1. You may move only one disk at a time.

2. No disk may ever rest atop a smaller disk. For example, if disk 3 is on a peg, then all disks below disk 3 must have numbers greater than 3.

How about two disks? How do you solve the problem when n = 2? You can do it in three steps. Here's what it looks like at the start:







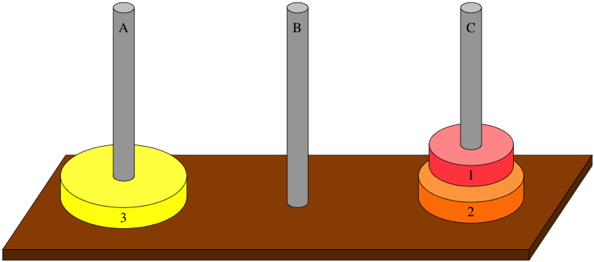
First, move disk 1 from peg A to peg C:

Notice that we're using peg C as a spare peg, a place to put disk 1 so that we can get at disk 2. Now that disk 2—the bottommost disk—is exposed, move it to peg B:

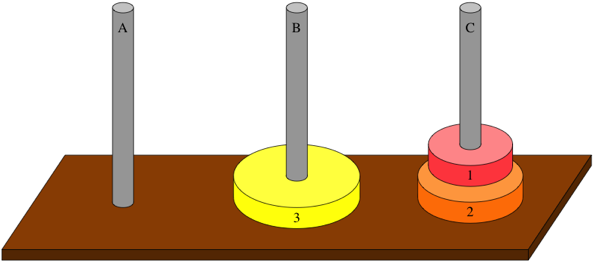
Finally, move disk 1 from peg C to peg B:

This solution takes three steps, and once again there's nothing special about moving the two disks from peg A to peg B. You can move them from peg B to peg C by using peg A as the spare peg: move disk 1 from peg B to peg A, then move disk 2 from peg B to peg C, and finish by moving disk 1 from peg A to peg C. Do you agree that you can move disks 1 and 2 from any peg to any peg in three steps? (Say "yes.")

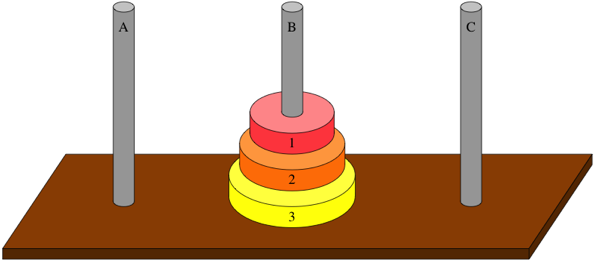
When you solved the Towers of Hanoi for three disks, you needed to expose the bottom disk, disk 3, so that you could move it from peg A to peg B. In order to expose disk 3, you needed to move disks 1 and 2 from peg A to the spare peg, which is peg C:



More to the point, by moving disks 1 and 2 from peg A to peg C, you have recursively solved a subproblem: move disks 1 through n-1(remember that n = 3) from peg A to peg C. Once you've solved this subproblem, you can move disk 3 from peg A to peg B:



Now, to finish up, you need to recursively solve the subproblem of moving disks 1 through n-1 from peg C to peg B. Again, you agreed that you can do so in three steps. (Move disk 1 from peg C to peg A; move disk 2 from peg C to peg B; move disk 1 from peg A to peg B.) And you're done:



And—you knew this question is coming—is there anything special about which pegs you moved disks 1 through 3 from and to? You could have moved them from any peg to any peg. For example, from peg B to peg C:

Recursively solve the subproblem of moving disks 1 and 2 from peg B to the spare peg, peg A. (Move disk 1 from peg B to peg C; move disk 2 from peg B to peg A; move disk 1 from peg C to peg A.)

Now that disk 3 is exposed on peg B, move to it peg C.

Recursively solve the subproblem of moving disks 1 and 2 from peg A to peg C. (Move disk 1 from peg A to peg B; move disk 2 from peg A to peg C; move disk 1 from peg B to peg C.)

No matter how you slice it, you can move disks 1 through 3 from any peg to any peg, moving disks seven times. Notice that you move disks three times for each of the two times that you recursively solve the subproblem of moving disks 1 and 2, plus one more move for disk 3.

How about when n = 4n? Because you can recursively solve the subproblem of moving disks 1 through 3 from any peg to any peg, you can solve the problem for n = 4n:

* Recursively solve the subproblem of moving disks 1 through 3 from peg A to peg C, moving disks seven times.
* Move disk 4 from peg A to peg B.
* Recursively solve the subproblem of moving disks 1 through 3 from peg C to peg B, moving disks seven times.

This solution moves disks 15 times (7 + 1 + 7) and, yes, you can move disks 1 through 4 from any peg to any peg.

At this point, you might have picked up two patterns. First, you can solve the Towers of Hanoi problem recursively. If n = 1. Otherwise, when n≥2, solve the problem in three steps:

* Recursively solve the subproblem of moving disks 1 through n-1 from whichever peg they start on to the spare peg.
* Move disk n from the peg it starts on to the peg it's supposed to end up on.
* Recursively solve the subproblem of moving disks 1 through n-1 from the spare peg to the peg they're supposed to end up on.

Second, solving a problem for nnn disks requires 2^n - 1. We've seen that it's true for:

