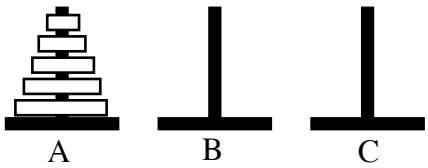
Towers of Hanoi Puzzle

(from *An introduction to Algorithms and Data Structures,* J. A. Storer, Springer, 2002)



Problem: You are given three posts labeled *A*, *B*, and *C*.

On Post *A* there are *n* rings of different sizes, in the order of the largest ring on the bottom to the smallest one on top.

Posts *B* and *C* are empty.

The object is to move the *n* rings from Post *A* to Post *B* by successively moving a ring from one post to another post that is empty or has a larger diameter ring on top.

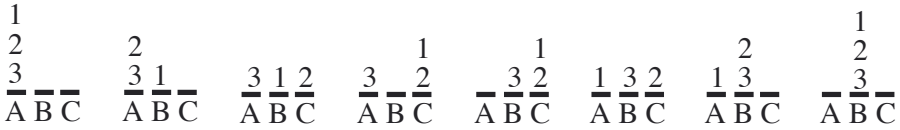
Inductive proof that the puzzle has a solution:

(*basis*) One ring can always be moved.

(*inductive step*) Assume that we can move *n*–1 rings from any given post to any other post. Since any of the rings 1 through *n*–1 can be placed on top of ring *n*, all *n* rings can be moved as follows:

1. Move *n*–1 rings from Post *A* to Post *C*.
2. Move ring *n* from Post *A* to Post *B*.
3. Move *n*–1 rings from Post *C* to Post *B*.

Example: The solution for 3 disks takes 7 steps:



Recursive algorithm: The proof by induction is constructive because it tells you how to solve a problem with *n* rings in terms of solutions to problems with *n*–1 rings; it corresponds directly to a recursive program. The procedure TOWER(*n*,*x*,*y*,*z*) moves *n* rings from Post *x* to Post *y* (using Post *z* as a "scratch" post):

```
procedure TOWER(n,x,y,z)
    if n>0 then begin
        TOWER(n–1,x,z,y)
        write "Move ring n from x to y."
        TOWER(n–1,z,y,x)
        end
    end
```

Complexity of TOWER

Recurrence relation for the number of moves: For n=1, the two calls for n–1 do nothing and exactly one move is made. For n>1, twice whatever the number of moves required for n–1 are made plus the move made by the *write* statement. Hence, the number of moves made by TOWER on input n is:

$$T(n) = \begin{cases} 1 \text{ if } n = 1 \\ 2T(n-1)+1 \text{ otherwise} \end{cases}$$

Theorem: For $n \geq 1$, *TOWER*(n,x,y,z) makes 2^n–1 moves.

Proof:

For n=1:

$$T(1) = 1 = 2^1 - 1$$

Now assume that TOWER works correctly for all values in the range 0 to n–1. Then:

$$\begin{aligned} T(n) &= 2T(n-1)+1 \\ &= 2(2^{n-1}-1)+1 \\ &= 2^n - 1 \end{aligned}$$

Time: The time for TOWER(n,x,y,z) is $\Theta(2^n)$ because it is proportional to the number of moves made. That is, every call to TOWER with $n \geq 1$ executes the write statement exactly once, and there are exactly two calls with n=0 for every call for n=1.

Space: The configuration of the rings is not explicitly stored (it is implicit in the reordering of x,y,z in the recursive calls). The total space used is dominated by the space for the recursion stack. Let c denote the space used by a copy of the local variables. Since the space used by the first call is released when it returns, the recurrence relation for the space used by TOWER(n,x,y,z) is:

$$S(n) = \begin{cases} c \text{ if } n = 1 \\ S(n-1) + c \text{ otherwise} \end{cases}$$

As we have already seen in the factorial example, the solution to this recurrence relation is $O(n)$.

Non-Recursive Towers of Hanoi Algorithm

Idea: If we unwind the recursion, it is not hard to see that the recursive Towers of Hanoi algorithm alternates between moving the smallest ring and one of the other rings, and that the smallest rings moves in a regular clockwise or counterclockwise fashion.

Lemma 1: In any minimal length solution to the Towers of Hanoi puzzle, the first and every other move is with the smallest ring.

Proof: Two consecutive moves with the smallest ring could be combined into a single move. Two consecutive moves of a larger ring must be from a peg back to itself (since the third peg must have the smallest peg on top), and hence could be eliminated.

Lemma 2: In any minimal length solution to the Towers of Hanoi puzzle, for odd n, the small ring always moves in a clockwise direction (A to B to C to A ...) and for even n in a counterclockwise direction (A to C to B to A ...).

Proof: The move in the TOWER procedure goes from Peg x to Peg y. For $n=1$, $x,y=A,B$ and the move is clockwise. If we inductively assume the lemma true for $n-1$ rings, then it follows for n by observing that both recursive calls reorder the pegs so that the relationship between the source and destination pegs (clockwise or counterclockwise) reverses.

Algorithm to move rings clockwise one post:

if n is odd **then** d := *clockwise* **else** d := *counterclockwise*

repeat

Move the smallest ring one post in direction d.

Make the only legal move that does not involve the smallest ring.

until all rings are on the same post

Time: $\Theta(2^n)$ — Exactly the same sequence of moves as the recursive version.

Space: $O(n)$ — Unlike the recursive version, the configuration of the rings must be explicitly represented; it suffices to use 3 stacks, each holding a maximum of n rings.

Observation: Both the mechanical translation of the recursive program to a non-recursive one and the "hand coded" version employ the stack data structure and use $O(n)$ space, although the constants in practice for both time and space are likely to be smaller for the hand-coded version.