1 | The Classical Towers of Hanoi

We can write the classical solution inductively:

1.1 | Base case

Move disc 1 to target peg

1.2 | Induction

- Move top n-1 discs to space peg
- Move the last disc to the target peg
- Move top n-1 stack back

You are essentially treating n-1 as one giant peg

1.3 | Recurrence Relation

We can write the number of moves via a recurrence relation:

- $T(1) = 1$ ("the base state takes 1 move")
- $T(n) = 2T(n-1) + 1$ ("move stack twice, also move the last disk")

This is inefficient to calculate. We will likely want to create a "closed-form solution". We will begin by making a bit of a table, and then prove that the eyeballed solution is correct.

2 | Evaluating Time Complexity, and "Divide and Conquer"

MergeSort's time complexity can be expressed as an open-form solution:

- $T(1) = 1$ ("it takes one operation to sort a list one one")
- $T(n) = 2T\left(\frac{2}{n}\right) + n - 1$ ("break in half, sort both sides, then merge")

We can simplify the expression, by defining $n - 1 = \theta(N)$.

- It takes $\theta(n)$ to combine the first two n/2 solutions
- It takes $2\theta(\frac{n}{2})$ to combine each of the two n/4 solutions
- ….eventually, the total number of levels is $log_2(n)$

At each level, there are n operations. In total, therefore, there is $nlog_2(n)$ operations.