Tower of Hanoi

Aditya Gupta

February 2023

1 Introduction

In this report, I talk about how I solved the classic problem of the Tower of Hanoi in Elixir.

2 Rules for the Tower of Hanoi problem

We need to abide by these rules when solving this problem:

1. One can only move one disk at a time, and only the top disk which can be located at any of the pegs.
2. A disk cannot be stacked on top of a smaller disk.

3 Approach

If we consider the case of just 2 disks, we can solve the problem in 3 steps by first moving the smallest disk to the auxiliary pole and then moving the largest disk to the destination pole followed by moving the auxiliary pole disk on top of the bigger disk. This approach can also be used for solving the case with more disks. One can think of the top n-1 disks as one disk. This way the problem is broken down to moving the n-1 disks to the auxiliary pole followed by moving the largest disk to the destination pole and then moving the other n-1 disks to the destination pole after that. We can deal with the problem of moving the n-1 disks using recursion as each case of moving n disks can be broken down into moving n-1 disks and 1 larger disk
Thus the above can be summarized using the following steps:

1. Move the top n-1 disks from the starting peg to the auxiliary peg.
2. Move the largest disk from the starting peg to the destination peg.
3. Move the disks from the auxiliary peg to the destination peg.

4 Implementation

```
def hanoi(0, _, _, _) do [] end
def hanoi(n, from, aux, to) do
    hanoi(n-1, from, to, aux) ++ # move tower of size n-1 ....
    [{:move, from, to}] ++ # [ move one disc ... ] ++
    hanoi(n-1, aux, from ,to) # move tower of size n-1 ...
end

# In the tests we denote the pegs using the atoms :a, :b and :c
# Moving a disk from one peg to the other is denoted using the
# tuple {:move, from, to}

# This is a function I wrote to print the number of moves corresponding
# to each n value
def moves (0) do "0 -> 0" end
def moves (n) do
    IO.inspect("#{n} -> #{length(Hanoi.hanoi(n,:a,:b,:c))}")
    moves(n-1)
end
```

5 Results

Here we notice that using our implementation we get $2^n - 1$ moves to be able to move the tower from the starting peg to the end peg where n is the number of disks.

Number of Disks	Number of Moves
0	0
1	1
2	3
3	7
4	15
5	31
6	63
7	127
8	255
9	511
10	1023