Towers of Hanoi

Axel Månson Lokrantz

Spring Term 2023

Introduction

The Towers of Hanoi is a mathematical puzzle consisting of three rods and a number of disks stacked on one rod of decreasing size. The largest at the bottom and the smallest at the top. The objective of the puzzle is the move the entire stack of disks to the last rod while following a given set of rules: One disk may be moved at a time. No disk may be placed on top of a disk that is smaller than it. A move consists of moving the upper disk of a rod and placing it on top of another stack or an empty rod.

The program was written in elixir, together with Axel Lystem at KTH.

The task

Solve for n disks

In this assignment we were asked to write a program to solve the puzzle of n disks and three pegs. The program should return the necessary sequence of moves to solve the tower. The pegs are represented as atoms `:a`, `:b`, `:c` and moves as tuples {`:move, to, from`}. The function takes four arguments, the number of disks, an auxiliary peg, the 'from' peg and the 'to' peg. The problem can be divided into smaller sub problems. Let's say we have have a stack of four disks. We start by moving the top 3 disks from rod 1 to rod 2 to be able to move the largest disk to rod 3. We now move the top 3 disks from rod 2 to rod 3. We came up with two implementations for the problem as seen below.

```
defmodule HanoiV1 do

  def hanoi(1, from, _, to) do [{:move, from, to}] end
  def hanoi(n, from, aux, to) do hanoi(n-1, from, to, aux) ++
    [{:move, from, to}] ++ hanoi(n-1, aux, from, to)
  end
end
```

Where the first line of code represent our base case of 1 disks and the second line the recursion.

```
defmodule HanoiV2 do

  def pm(start, finish) do
    IO.puts("#{start} -> #{finish}")
  end
  def h(n, start, finish) do
    case n do
      1 -> pm(start, finish)
      _ ->
        other = 6 - (start + finish)
        h(n-1, start, other)
        pm(start, finish)
        h(n-1, other, finish)
    end
  end
end
```

The two programs are fairly similar. HanoiV2 does not use an auxiliary rod as input while HanoiV1 does. However, the recursion is the same for both solutions. Since there are two recursion calls in the same function the code gets quite tricky to follow, therefor we made an image that hopefully makes it a little more clear.

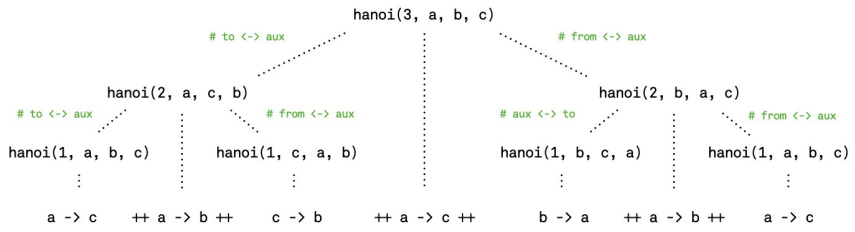

Figure 1: Recursion steps for HanoiV1.

The example is made with a puzzle of three disks since the number of moves rapidly increase the more disks you have. `hanoi(:3, a, b, c)`} will result in the following sequence of moves.

{:move, :a, :c},
{:move, :a, :b},

{:move, :c, :b},
{:move, :a, :c},
{:move, :b, :a},
{:move, :b, :c},
{:move, :a, :c}

This is equal to the example that was given in the product assignment, and is tested to be correct.