Towers of Hanoi

Nariman Haidar

February 10, 2023

1 Introduction:

In this assignment we are going to solve the problem of Tower of Hanoi puzzle, and to do that we will use recursive strategy to solve this problem.

2 The towers:

We'll talk a little bit about Tower of Hanoi in this section. Three rods and numerous disks are used in this mathematical puzzle, which is called the Tower of Hanoi. On one rod, the disks are originally arranged in an ascending cone-shaped tower with respect to size.

The goal of the problem is to slide every disk in the stack to the final rod while adhering to a set of rules:

1. One disk at a time can be moved.
2. A disk that is smaller than it cannot be stacked on top of another disk.

3 Recursive strategy:

We will discuss the Tower of Hanoi solution strategy in this section, taking into account all of the cases we currently have. In the base case that when the number of disks is zero that we do not need to move any thing and that is mean zero movement and in that case we return an empty list.

Now if we have a number of disks that are in the first peg a and we will to move all disks to peg c and we have temporary peg (b):

1. First we should move all of the disks aside from the final one to the temporary peg (`hanoi(n, a, c, b)`) and that is mean (`hanoi(number_of_disks ,source(a) ,goal(c),temp(b))`).
2. Then the final ring from the starting peg should then be moved to the goal peg (`[{:move, a, c}]`) and that is mean (`[{:move, from source(a), to goal(c)}]`).
3. Then Use the same procedure to transfer the other rings from the temporary peg to the desired peg (`hanoi(n, b, a, c)`) and that is mean (`hanoi(n, a, c, b)`) and that is mean (`hanoi(number_of_disks ,temp(b) ,source(a) ,goal(c))`).

To determine the smallest number of moves needed to solve the puzzle for n number of discs, the resulting program employs a recursive solution.

```
defmodule Hanoi do
    def hanoi(0, _, _, _) do [] end
    def hanoi(n, a, b, c) do
        hanoi(n-1, a, c, b) ++
        [{:move, a, c}] ++
        hanoi(n-1, b, a, c)
    end
end
```

4 The sequence:

The function that returns the series of movements required to solve towers of Hanoi puzzles with a given size n will be implemented in this section. Three pegs (a, b, and c) make up our system, and a move is represented by the tuple:
(`[{:move, from, to}]`).

If we want three disks we have to run this command to get the result `Hanoi.hanoi(3, :a, :b, :c)`, and in the assignment pdf we can see the result of this command, but if we want the result for four disks we have to run this command to get the result `Hanoi.hanoi(4, :a, :b, :c)` and the result is:

```
[
  {:move, :a, :b},
  {:move, :a, :c},
  {:move, :b, :c},
  {:move, :a, :b},
  {:move, :c, :a},
  {:move, :c, :b},
  {:move, :a, :b},
  {:move, :a, :c},
  {:move, :b, :c},
  {:move, :b, :a},
  {:move, :c, :a},
  {:move, :b, :c},
  {:move, :a, :b},
  {:move, :a, :c},
  {:move, :b, :c}
]
```

The total of moves $= 2^n - 1$. This gives us in case n = 3 the moves are 7 and in the case of n = 4 the moves are 15 and in the case of n = 10 the moves are 1023.