Towers of Hanoi

Sareh Jalalizad

Spring Term 2023

Introduction

For this assignment, we were supposed to implement a program to solve the Tower of Hanoi puzzle through the use of recursion. To solve the puzzle, we needed to use the recursive approach, which involves breaking down the problem into smaller sub-problems and solving them individually, until the base case is reached. This strategy provides a simple solution to the Tower of Hanoi puzzle.

The towers of Hanoi

The Tower of Hanoi is a classic puzzle consisting of three pegs and a number of discs of different sizes, which can slide onto any peg. The objective is to move the entire stack to another peg, obeying the following simple rules:

- Only one disc can be moved at a time.
- Each move consists of taking the upper disc from one of the stacks and placing it on top of another stack or on an empty peg.
- No disc may be placed on top of a smaller disc.

For this task, we should write a program to solve this puzzle with n discs. The goal is to determine the sequence of moves needed to solve it. The pegs are called ":a," ":b," and ":c," and each move is recorded as a tuple.

```
{:move, from, to}
```

The function requires four parameters: the number of discs, the starting peg, an auxiliary peg, and the target peg where the tower should ultimately be placed. When implementing a recursive function, it's important to handle the base case correctly. In this puzzle, the base case is solving the tower of Hanoi for a tower with 0 discs.

Next, the recursive step involves solving the puzzle for n discs. The first step is to move the n-1 discs from the starting peg to the auxiliary peg, using the target peg as the intermediate peg. This is needed to clear the way for the largest disc to be transferred to the target peg. Finally, the remaining n-1 discs from the auxiliary peg to the target peg, using the starting peg as the intermediate peg. This process is repeated until we have solved the puzzle for all n discs, with the largest disc ultimately being moved to the target peg.

```
def hanoi(0,_,_,_) do [] end
def hanoi(n, from, aux, to) do
  hanoi(n-1, from, to, aux) ++ [{:move, from, to}]
  ++ hanoi(n-1, aux, from, to)
end
```

When we call the Hanoi function for 3 discs, it produces the same result as the example provided in the assignment instructions. To solve the tower of Hanoi puzzle with 4 discs, hanoi(4, :a, :b, :c)), a total of 15 moves are required, as can be seen from the example output.

```
[
  {: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}
]
```

Accordingly, It is possible to calculate the number of moves needed to solve the tower of Hanoi for a given number of discs using the formula $2^n - 1$. For example, if we were to solve the tower of Hanoi puzzle with 10 discs, we would need a total of $2^{10} - 1 = 1023$ moves.

Conslusion

Overall, this assignment was interesting, and the tower of Hanoi puzzle provides a simple yet effective demonstration of the power of recursion in solving problems.