Hands-on Activity 4.1: The Tower of Hanoi Problem

Intended Learning Outcomes

- Analyze and solve computational problems using problem solving methodologies and decomposition

Objectives

- Introduce students to the Tower of Hanoi Problem;
- Apply the fundamentals of logical and algorithmic thinking;
- Solve the Tower of Hanoi Problem using Python.

Discussion

The Tower of Hanoi is a mathematical puzzle invented by the French mathematician Edouard LucasLinks to an external site. in 1883.

There are three pegs, source(A), Auxiliary (B) and Destination(C). Peg A contains a set of disks stacked to resemble a tower, with the largest disk at the bottom and the smallest disk at the top. figure 1 Illustrate the initial configuration of the pegs for 3 disks. The objective is to transfer the entire tower of disks in peg A to peg C maintaining the same order of the disks.

Obeying the following rules:

Only one disk can be transfer at a time. Each move consists of taking the upper disk from one of the peg and placing it on the top of another peg i.e. a disk can only be moved if it is the uppermost disk of the peg. Never a larger disk is placed on a smaller disk during the transfer. The solution to the puzzle calls for an application of recursive functions and recurrence relations.

```
TOH( n,  Sour, Aux , Des)
If(n=1)
    Write ("Move Disk “, n ," from ", Sour ," to ",Des)
Else
    TOH(n-1,Sour,Des,Aux);
    Write ("Move Disk “, n ," from ", Sour ," to ",Des)
    TOH(n-1,Aux,Sour,Des);
END
```

Procedure

1. Implement the algorithm for the Tower of Hanoi Problem as shown above.
2. Write your observations for your written algorithm/solution.

Tower of Hanoi problem using Dynamic Programming (Bottom-up Approach)

```
class TOH:
    def __init__(self, n_disk):
        self.n_disk = n_disk
        self.move = []

    def moveDisk(self, disc, from_rod, to_rod):
        print(f"Move disk {disc} from {from_rod} to {to_rod}")
        self.move.append((disc, from_rod, to_rod))

    def stateDisk(self):
        self.solveTOH(self.n_disk, "Source", "Destination", "Auxiliary")

    def solveTOH(self, n, sour, des, aux):
        if n == 1:
            self.moveDisk(1, sour, des)
            return
```

```
            self.solveTOH(n - 1, sour, aux, des)
            self.moveDisk(n, sour, des)
            self.solveTOH(n - 1, aux, des, sour)

        def getMove(self):
            return self.move

    TowerOfHanoi = TOH(3)
    TowerOfHanoi.stateDisk()
```

```
Move disk 1 from Source to Destination
Move disk 2 from Source to Auxiliary
Move disk 1 from Destination to Auxiliary
Move disk 3 from Source to Destination
Move disk 1 from Auxiliary to Source
Move disk 2 from Auxiliary to Destination
Move disk 1 from Source to Destination
```

Observation

This code implements the given algorithm in the activity. All steps are correct, this is based on the given figure with 7 steps with n = 3. The bottom-up approach was used in this code to solve the Tower of Hanoi problem. I think this method can be a solution for the algorithm. I have implemented in the code the conditions when moving the disk.

Supplementary Activity

1. Explain the programming paradigms/techniques (like recursion or dynamic programming) that you used to solve the given problem.
2. Provide screenshots of the techniques and provide a quick analysis.

Quick Analysis

For my analysis, I have used OOP approach and dynamic programming specifically bottom-up approach for solving the given problem because the subproblems must be breakdown to find the best solution which is the disk needed to be transfer one at a time and avoid redundant solutions. It also uses recursion because the disk will be transfer in other state. In my code, I have implemented the algorithm in the condition to have a looping statement.