

✓ 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.

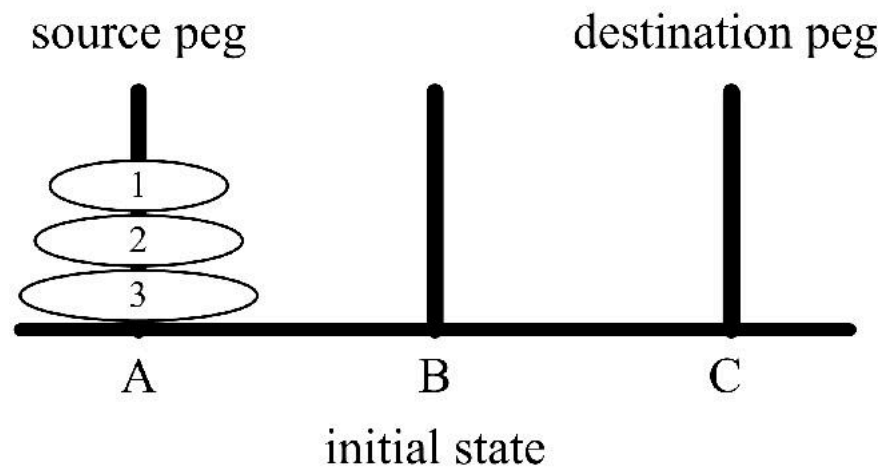


Figure 1.1 Initial State of the Tower of Hanoi Problem

A skeletal recursive procedure (Outline) for the solution of the problem for N number of disks is as follows:

Move the top N-1 disks from peg A to peg B (using C as an auxiliarypeg) Move the bottom disk from peg A to peg C Move N-1 disks from Peg B to Peg C (using Peg A as an auxiliary peg) The pictorial representation of the skeletal recursive procedure for N=4 disks is shown in Figure 2.

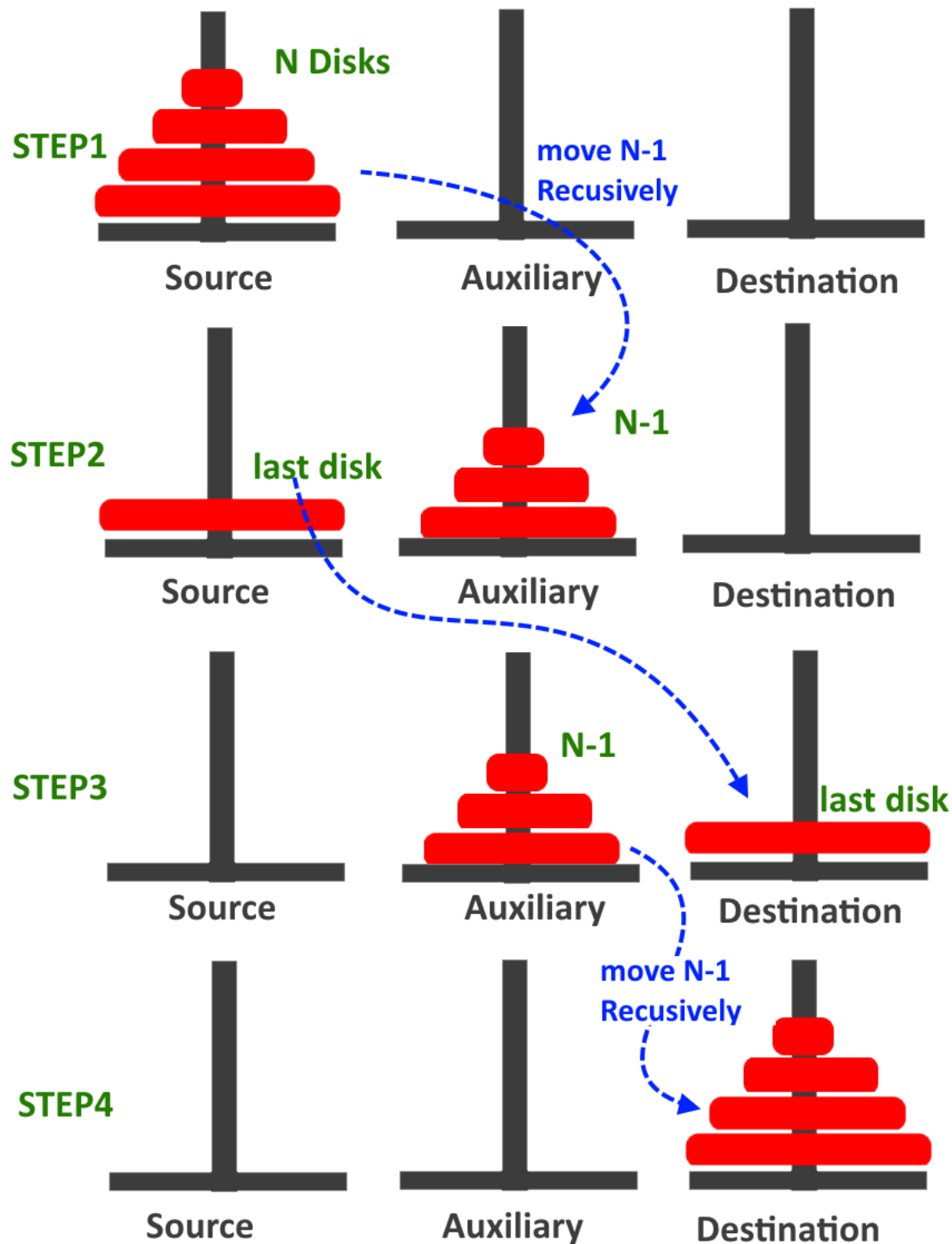
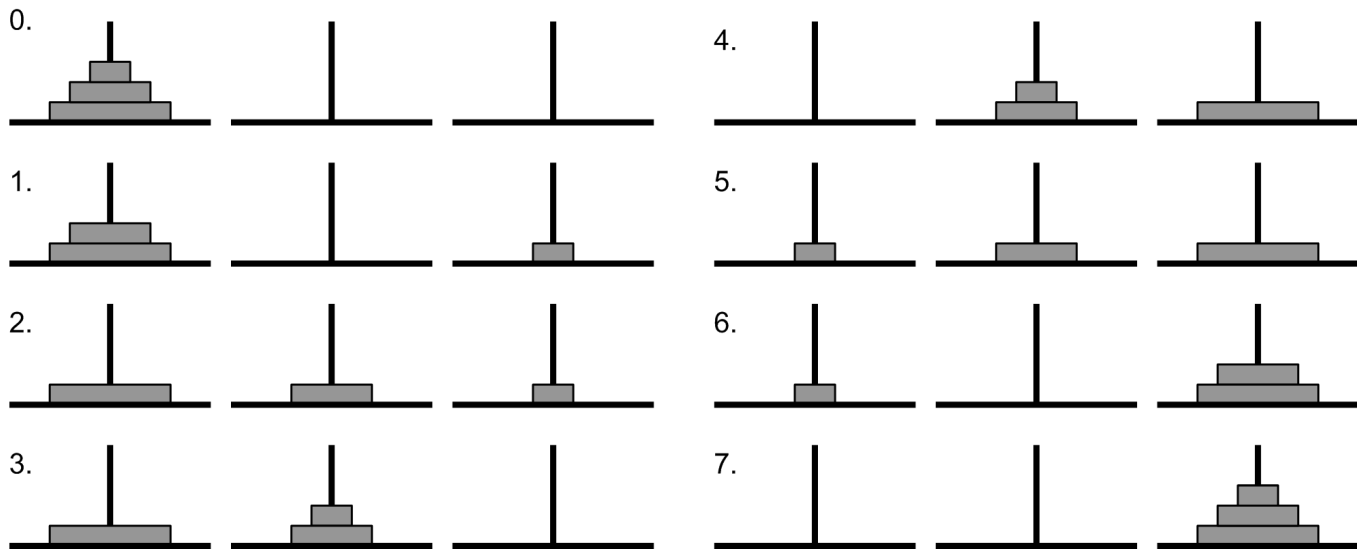


Figure 2 Tower of Hanoi Step-by-Step

Algorithm

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 Let's take an example to better understand the algorithm (For $n=3$).



Procedure

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

Supplementary Activity

Explain the programming paradigms/techniques (like recursion or dynamic programming) that you used to solve the given problem. Provide screenshots of the techniques and provide a quick analysis. Your submission must be the link to the notebook on your Github account.

```
n = int(input("Enter number of disks: ")) # get number of disks
```

```
def TowerofHanoi(n, Sour, Aux, Des):
    if n == 1 :
        print(f"Move disk number {n} from {Sour} to {Des}") # output for moving disk number 1 or smallest disk
        return

    TowerofHanoi(n-1, Sour, Des, Aux) # next changing of source and destination pegs
    print(f"Move disk number {n} from {Sour} to {Des}") # output for moving disk number 1 or smallest disk
    TowerofHanoi(n-1, Aux, Sour, Des) # next changing of source and destination pegs
TowerofHanoi(n, 'A', 'B', 'C')
```

```
Enter number of disks: 3
Move disk number 1 from A to C
Move disk number 2 from A to B
Move disk number 1 from C to B
Move disk number 3 from A to C
Move disk number 1 from B to A
Move disk number 2 from B to C
Move disk number 1 from A to C
```

```
n = int(input("Enter number of disks: ")) # get number of disks
```

```
def TowerofHanoi(n, Sour, Aux, Des):
    if n == 1 :
        print(f"Move disk number {n} from {Sour} to {Des}") # output for moving disk number 1 or smallest disk
        return

    TowerofHanoi(n-1, Sour, Des, Aux) # next changing of source and destination pegs
    print(f"Move disk number {n} from {Sour} to {Des}") # output for moving disk number 1 or smallest disk
    TowerofHanoi(n-1, Aux, Sour, Des) # next changing of source and destination pegs
TowerofHanoi(n, 'A', 'B', 'C')
```

```
Enter number of disks: 4
Move disk number 1 from A to B
Move disk number 2 from A to C
Move disk number 1 from B to C
Move disk number 3 from A to B
Move disk number 1 from C to A
Move disk number 2 from C to B
Move disk number 1 from A to B
Move disk number 4 from A to C
Move disk number 1 from B to C
Move disk number 2 from B to A
Move disk number 1 from C to A
```

```

Move disk number 3 from B to C
Move disk number 1 from A to B
Move disk number 2 from A to C
Move disk number 1 from B to C

```

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);

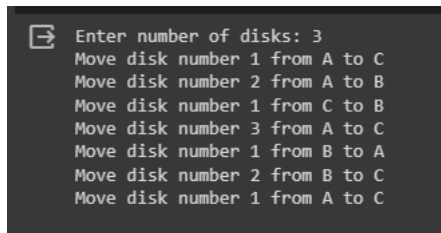
```

END

In basic knowledge, the Tower of Hanoi follows a certain algorithm. As provided, if $n == 1$, it will move it from source peg to destination peg. As you go on, the source, auxiliary, and destination pegs change different roles accordingly by considering the destination peg as the new auxiliary peg and vice versa. Now to place the smallest disk to its destination peg, TowerofHanoi($n-1$, Aux, Sour, Des) is implemented for it to be consequently followed by the if statement again. These conditions are repeated until the recursion process terminates. It is noticeable that this is recursion programming as you must return the values in order to temporarily store their positionings.

✓ Supplementary Activity

While implementing the algorithm, a pattern could be derived. Notice that the smallest disk is always moved from peg to peg after moving any



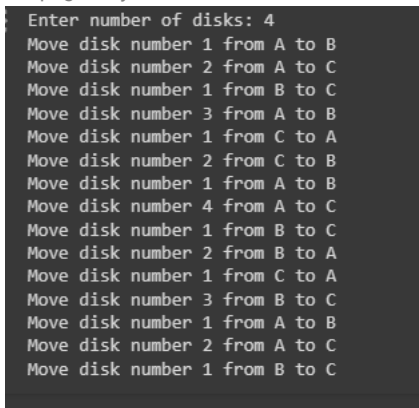
```

Enter number of disks: 3
Move disk number 1 from A to C
Move disk number 2 from A to B
Move disk number 1 from C to B
Move disk number 3 from A to C
Move disk number 1 from B to A
Move disk number 2 from B to C
Move disk number 1 from A to C

```

other peg. Another pattern derived is the consecutive positions of the pegs.

As you can see, every time disk 1 is moved from peg to peg. It creates a pattern in which it is A to C, C to B, and B to A causing the destination peg to be the source peg everytime the smallest disk is moved. For any other disk, A to B, A to C, and B to C are the used consecutive peg-to-



```

Enter number of disks: 4
Move disk number 1 from A to B
Move disk number 2 from A to C
Move disk number 1 from B to C
Move disk number 3 from A to B
Move disk number 1 from C to A
Move disk number 2 from C to B
Move disk number 1 from A to B
Move disk number 4 from A to C
Move disk number 1 from B to C
Move disk number 2 from B to A
Move disk number 1 from C to A
Move disk number 3 from B to C
Move disk number 1 from A to B
Move disk number 2 from A to C
Move disk number 1 from B to C

```

peg movements.

As for 4 disks, the smallest disk's movement is moved consecutively based on the position of the pegs which is A to B, B to C, and C to A. Each size of disk has its own pattern. Another example is for disk 2 which for every next movement the destination peg becomes the source peg. That is why I could come up with a conclusion that this Tower of Hanoi problem in the end does not require manual thinking which why it could be solved using a certain algorithm because of its derived patterns.

