Computer Science II
Project 1: Hanoi Tower

TAERAKUL Janat, #19B60096

December 25, 2019

1 Hanoi Tower

The "Tower of Hanoi" is a mathematical puzzle consists of disks with different sizes and three rods as shown in figure 1. The objective of the game is to move the tower of disks from one rod to another. There are several rules that must be followed when moving disks:

- Only one disk can be moved at a time.
- A disk can only be placed on top of a disk larger than itself.

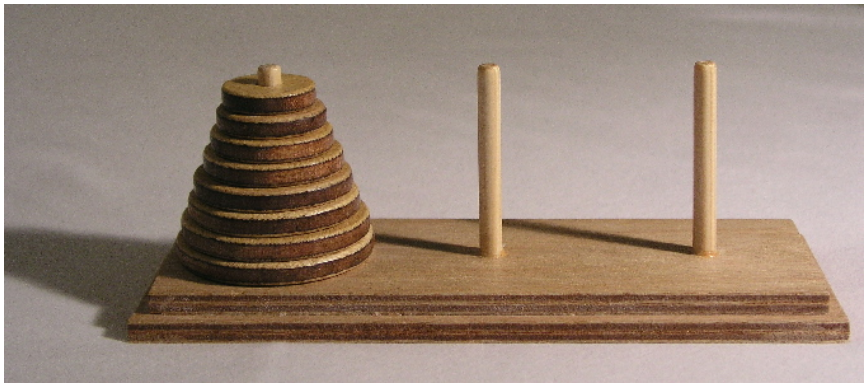

Figure 1: The Tower of Hanoi

This program (`hanoi.py`) takes an integer `n` for the number of disks as an input and assume there is a tower of `n` disks at the peg `0`. Then, it prints

2 How the Program Works

2.1 Global Variables

1. `n` - Then number of disks in the tower of Hanoi.
2. `step` - The counter for steps of moving disks.
3. `block` - A character for printing the disks.

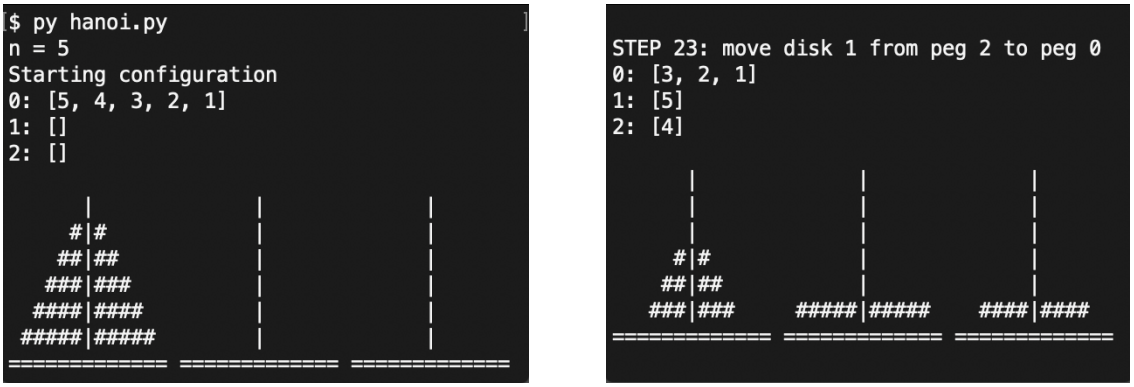

Figure 2: The display of running program

2.2 How to Move Tower

The function `move_tower()` is a recursive function. The base case or the smallest case for the function is when the tower is a disk. In that case, the function would just simply moves the disk by calling function `move_disk()` to move the disk from source peg to destination peg. For the recursive part, the function considers the tower of `nb_disk` disks as a tower of `nb_disks-1` disks and a disk of size `nb_disks`

```
def move_tower(pegs, nb_disk, source, dest):
    spare = 3 - source - dest
    if nb_disk == 1:
        move_disk(pegs, source, dest)
    else:
        move_tower(pegs, nb_disk - 1, source, spare)
        move_disk(pegs, source, dest)
        move_tower(pegs, nb_disk - 1, spare, dest)
```

3 Display Improvement

In the template code, there was no step numerating, so I created a global variable named `step` and used is as a step counter. I added two lines of codes and changed printing format in the `move_disk()` function as shown in code below.

```
global step
step += 1
print(f"STEP {step}: move disk {disk} from peg {source} to peg {dest}")
```

display_tower(pegs) - Input data of `pegs` into this function, the it prints the display of towers as shown in figure 2.

In addition to the example, lists-printing, display, I added the illustration of the three pegs and disks by using similar principal to the Project 1 of Computer Science course. I used loops and simple calculation for printing the game's configuration.

In detail, there is a loop running through each line of the figure and another loop, nested inside, running through the three pegs. For each line of each peg, it will find if there is a disk to be printed or not. If there is a no disk to be printed, the value of `size`,

which denotes the size of the disk, will be 0. If there is a disk to be printed, the value of `size` will be the size of the disk.

After the program knows the size of the disk that has to be printed, it will print the disk out by printing *n-size+1* spaces (the width of the pegs is one block bigger than the biggest disk) and `size` blocks. Then, the program prints a pole (`|`) follows with `size` blocks and *n-size+2* spaces (add one space for a space between pegs).

Lastly, the program prints (`=`) as the bases of the pegs by simply loops three times for three pegs and prints `2*n+3` times of `=` for each time.

```
def display_tower(pegs):
    for height in range(n + 1, 0, -1):
        for peg in pegs:
            if height <= len(peg):
                size = peg[height-1]
            else:
                size = 0

            for k in range(n - size + 1):
                print(' ', end = '')
            for k in range(size):
                print(block, end = '')
            print('|', end = '')
            for k in range(size):
                print(block, end = '')
            for k in range(n - size + 2):
                print(' ', end = '')
        print()

    for i in range(3):
        for j in range(2 * n + 3):
            print('=', end = '')
        print(' ', end = '')

    print()
    print()
```