COMPUTER SCIENCE II

PROJECT 1 - TOWER OF HANOI (ANIMATION)

December 20, 2018

Student ID: 18B09790

Tokyo Institute of Technology

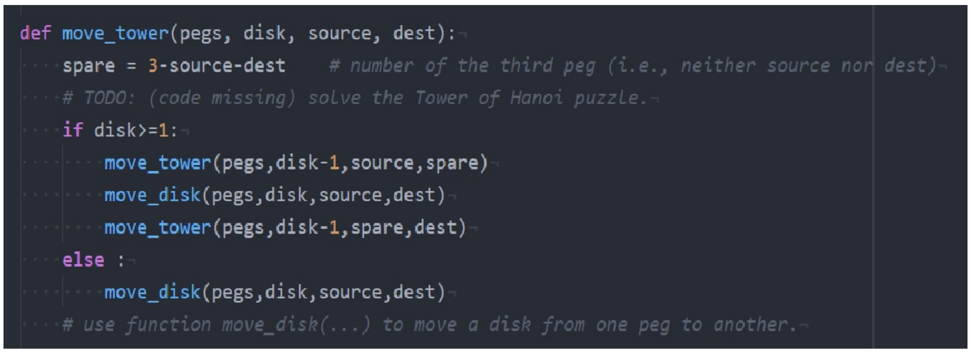
INTRODUCTION

This program is about solving tower of hanoi problem step by step in animation. By using the program given by professor in the class I modified it to make the display easier to understand. The main technique of this program is creating a recursive function which i believe that it is the very optimized method for solving this problem. Although the display part might take a lot of time.

WHAT I ADDED

The parts i added to the program are the recursive algorithm which will complete the program and the other part I added is the animation display part. I also delete some error-check part which I found it burdening the program after complete the program (It is very useful for debugging during writing the program though).

Recursive algorithm



```python
def move_tower(pegs, disk, source, dest):
    spare = 3-source-dest    # number of the third peg (i.e., neither source nor dest)
    # TODO: (code missing) solve the Tower of Hanoi puzzle.
    if disk>=1:
        move_tower(pegs,disk-1,source,spare)
        move_disk(pegs,disk,source,dest)
        move_tower(pegs,disk-1,spare,dest)
    else :
        move_disk(pegs,disk,source,dest)
    # use function move_disk(...) to move a disk from one peg to another.
```

Figure 1: move-tower function

The main idea of the first step of the program is to move the biggest disk to the target peg. To achieve this we have to move all the upper disk to the spare peg and then move the biggest disk so called the base disk from the source peg to the target peg. Thus earlier step is to move the n-1 size tower (tower whose disk sizes are from 1 to n-1) to the spare peg, thus the "spare peg" now will considered to be the destination peg for the smaller tower. By keeping doing this idea from n sized tower until 2 sized tower (using recursive function), the first actual step will be to move the smallest disk to its spare peg, and then move the second smallest disk to the other peg. Then to make the 2 disk-tower, bring back the smallest disk from the spare peg to the other peg creating the tower. After that, we can bring the next base disk to the empty

peg and then move the 2 disk tower back on top of it creating 3 disk-tower. By keep doing so, we will be able to move the complete n sized tower to the other peg.
Simply put, to implement this in the program, the only part needed to be added is the move-tower function like Figure 1. First, we make recursive function from the bigger tower until the program consider moving the smallest disk as the tower like the 5th line in the figure(78th line referring to the code). After that I move the base disk to the empty peg like the 6th line. Then I move the smaller tower back on the base disk like the 7th line. By keep doing this, the program will be able to complete the task. Noted that if-else written in the function is to separate the case of moving 1 disk-sized tower which is considered as moving the smallest disk (like the 9th line) from the other case.

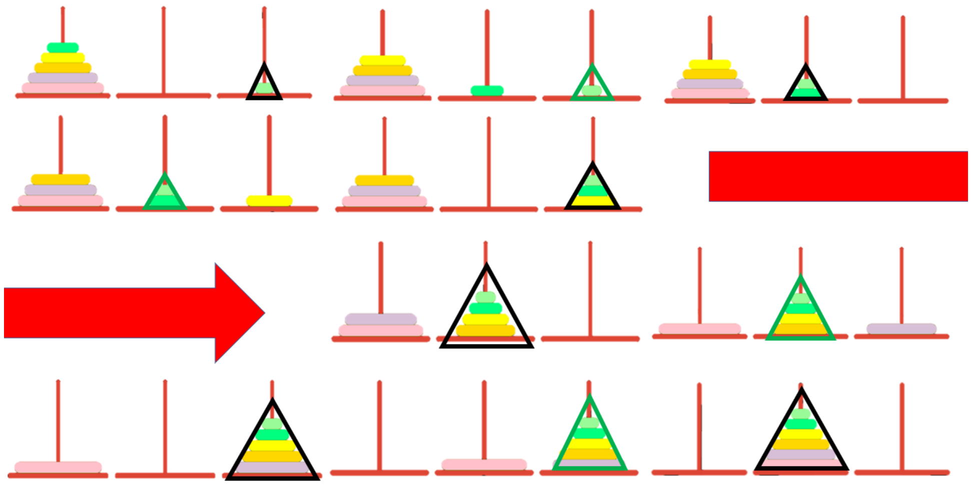
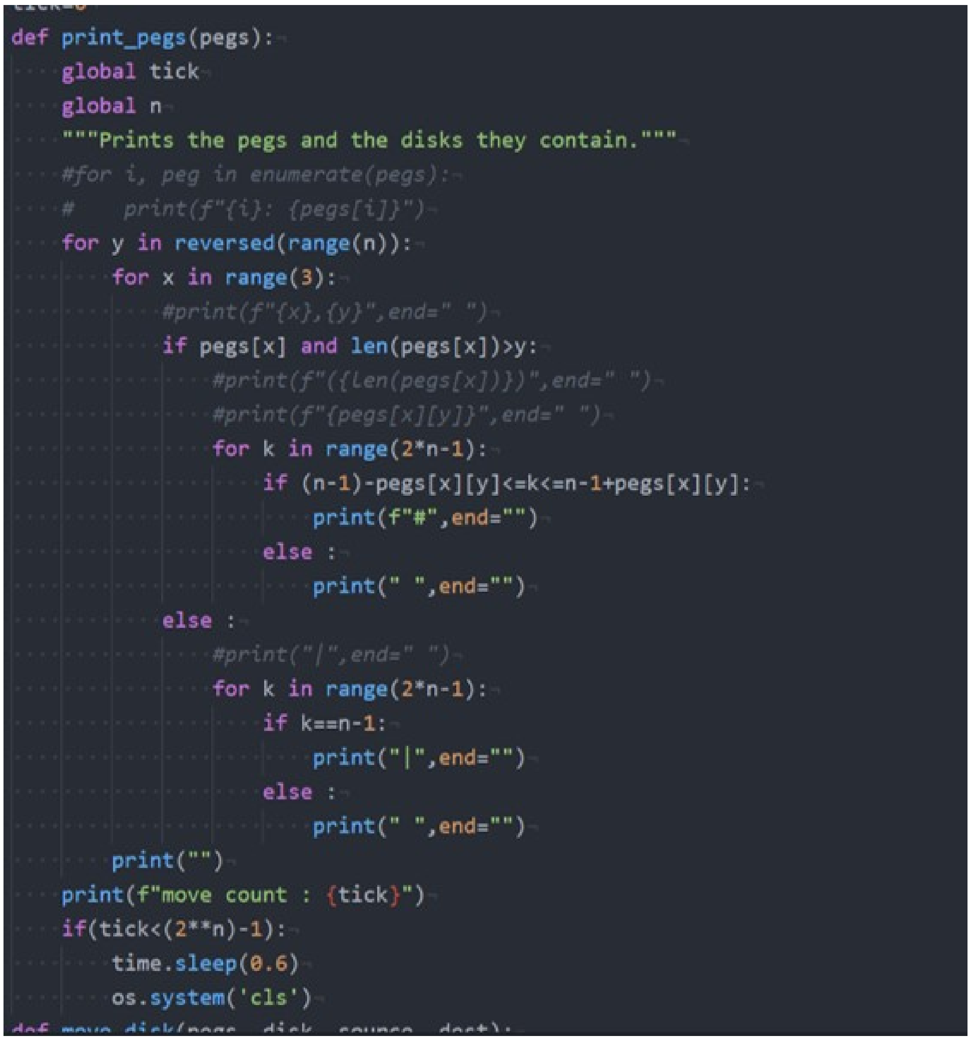


Figure 2: step by step explanation

From Figure 2, Black triangle is considered as moved smaller tower which represent the 5th line step from Figure 1. Between each images in the figure the base disk is being move to the empty peg which represents the 6th line. Last, the dark green triangle represents moving the smaller tower back to the new base disk like what the 7th line do.

Animation display

This part, I modified mainly the print-pegs function to create animation of moving tower from one to the other whereas size of the disks is also displayed. I used pegs which is a 2 dimension array to represent coordinate of each point/disk. So in this case the coordinate of x axis will be 0,1,2 represent peg 1, peg 2 and peg 3. For y axis, the coordinate will be 0,1,...,n-1 equal to the height of the highest possible tower. By running loop for all coordinate and apply some condition to the display process as Figure 3, I able to create the good display of the tower. Regarding the condition I apply in the loop, it is to check whether there is a disk at the current coordinate or not, if there is no disk, the program will print out the stick(peg). Otherwise the program will print of the disk whose size is calculated by 2*n-1. For example the smallest disk's size will be 1, the second smallest disk's size will be 3 and so on. I also add time.sleep function for 0.6 second and clear screen function inside printpegs function to clear the screen every time there is a movement. I also check whether is this is the last move or not since I don't want to clear screen after the last move. Furthermore, I also add clear screen function after the program receive input too so that the input and other command in the command line got wipe out before the animation start. Noted that i include time and os library to the program in order to use those function(clear screen and time sleep function).



```python
def print_pegs(pegs):
    global tick
    global n
    """Prints the pegs and the disks they contain."""
    #for i, peg in enumerate(pegs):
    #    print(f"{i}: {pegs[i]}")
    for y in reversed(range(n)):
        for x in range(3):
            #print(f"{x},{y}",end=" ")
            if pegs[x] and len(pegs[x])>y:
                #print(f"({Len(pegs[x])})",end=" ")
                #print(f"{pegs[x][y]}",end=" ")
                for k in range(2*n-1):
                    if (n-1)-pegs[x][y]<=k<=n-1+pegs[x][y]:
                        print(f"#",end="")
                    else :
                        print(" ",end="")
            else :
                #print("|",end=" ")
                for k in range(2*n-1):
                    if k==n-1:
                        print("|",end="")
                    else :
                        print(" ",end="")
        print("")
    print(f"move count : {tick}")
    if(tick<(2**n)-1):
        time.sleep(0.6)
        os.system('cls')
```

Figure 3: print-pegs function