CS5001 Project 6 Report

Shang Xiao

March 2, 2022

1 Problem Description

For final project planning, I have decided to implement the Tower of Hanoi game. In this game, users can choose to play the game by themselves, or they can choose to see the minimum number of steps (solutions) directly. Of course, there will be hints and instructions provided for the user to make the game more playable. The program will also differentiate between the user's number of steps and the minimum number of steps once the user completes the game.

I am planning to solve two significant problems in this project:

Firstly, we need to allow the user to make mistakes. We want to let users to play the game by themselves and perform incorrect steps. Then, the program will prompt the user if they have committed a false step and what they should do next to continue the game. When the user completes the game, the program calculates the total steps that users have taken. At the same time, the user will be prompted for the minimum number of steps as the optimal solution for this game. By comparing user steps with the optimal solution, the user can tell whether they have completed the game well.

Secondly, we need to provide an optimal solution to solve the problem with a minimum number of steps so that the user can refer to this solution when they have encountered difficulties or got stuck while completing the game.

2 Background Information

Tower of Hanoi is a very classic game. In this game, there are three pillars and a given number of disks of different sizes. In the beginning, all disks are placed on the first pillar ordered from the largest at the bottom to the smallest at the top, as shown below:

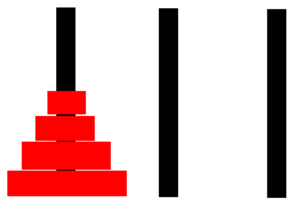


Figure 1: Initial setup

Users can take the top disk from the first pillar and place it onto another pillar within one move. However, users can not place the larger disk on top of the smaller ones. As shown below:

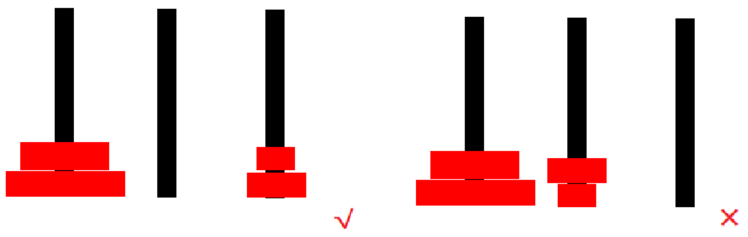
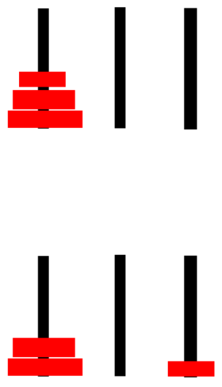
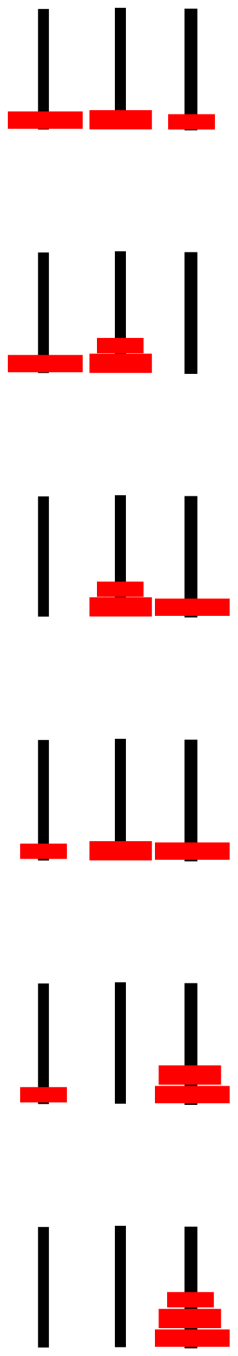


Figure 2: Example of one move

The goal is to move all disks onto the third pillar.

The optimal solution to a Tower of Hanoi problem with 3 disks is given below:





Through this example, we can clearly see the strategy used by the user:

There are three disks: the largest, the second-largest, and the smallest.

Move the smallest disk onto the third pillar and move the second-largest disk onto the second pillar.

Move the smallest disk onto the second-largest disk and move the largest disk onto the third pillar.

Move the smallest disk onto the first pillar and move the second-largest disk onto the largest disk.

Move the smallest disk onto the second-largest disk. Done!

3 List of Challenges

1. Knowledge in recursion is required to solve this problem. So far, we have learned about functions and loops, but we have yet to explore recursion.

2. We need to make sure that the user can play the game. For every move, we need to store data such as the size of each disk and the pillar that each disk has been moved onto.

3. Knowledge in exception handling is also required to solve this problem. There will always be cases that users do not follow instructions. For example: moving a larger disk onto a smaller disk.

4 List of Problems I Can Solve

1. We can create a menu that allows the user to elect whether they want to see the optimal solution first. For example:

```python
print('Please chose to start a game or check the answer')
print('1. Start a game')
print('2. Check the answer')
choice = int(input())
if choice == 1:
    start_game()
else:
    check_answer()
```

2. Record user solution and compare with the optimal solution, then return feedback to the user. Since a game of n disks needs at least $2^n - 1$ steps to be solved, we can limit the steps. If user steps are not close to $(2^n - 1)$, return solution is not optimal. Otherwise, return good feedback. Note that there is an "optimal multiplier of 1.x" implied in the example below because we want to make the game more fun. So that even the user cannot find the optimal solution, if they were close, we also want to return good feedback.

```python
if step > (2 ** n - 1) * 1.2:
    print('Not optimal')
else:
    print('You are really close to the optimal solution!')
```

3. To check if a user has completed the game, we can implement the following *while* loop:

```python
while not check_win():
    continue
```

5 Acknowledgements

5.1 Website consulted (Information about the final project):

https://mathworld.wolfram.com/TowerofHanoi.html/ - Definition and some basic information of Tower of Hanoi.

https://www.geeksforgeeks.org/c-program-for-tower-of-hanoi/ - Solution of Tower of Hanoi.