Advanced Topics Chinese Ring Puzzle (5 Rings) Project

The Chinese Ring Puzzle

INTRODUCTION

The Chinese Ring Puzzle, otherwise known as the nine linked rings puzzle or *jiu lian huan* 九连环, is perhaps among the greatest mechanical inventions from China.

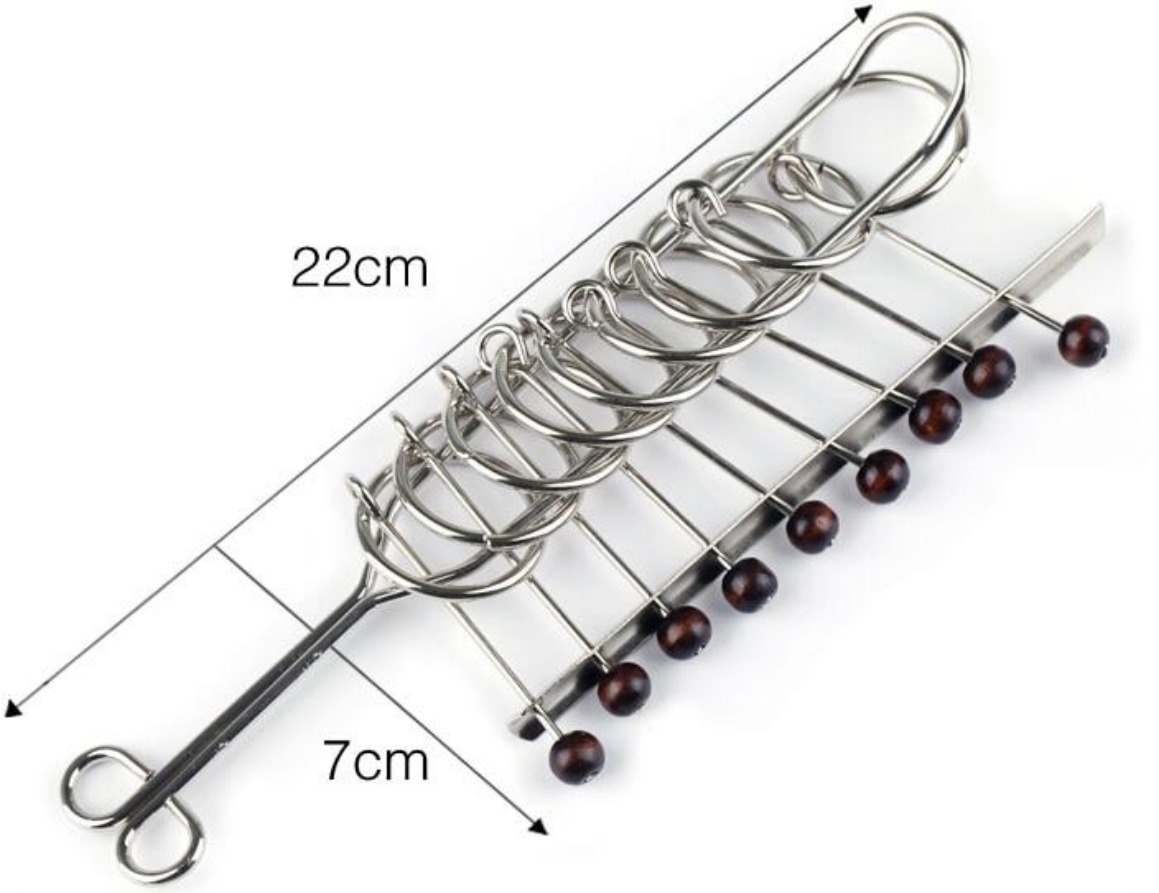

Figure 1.1 Chinese Ring Puzzle

As shown in Figure 1.1, the puzzle consists of a long loop interlocked with nine rings. The objective is to disentangle all nine rings from the loop. There are several ways of approaching this problem, but generally, the solution is laborious and requires a lot of patience. However, once one understands the pattern, the Chinese Ring Puzzle can be a very interesting exercise.

Our Solution

We combined piecewise functions and Python to construct a computer program that prints a binary step-by-step manual for solving the Chinese Ring Puzzle.

Advanced Topics Chinese Ring Puzzle (5 Rings) Project

SOLUTION

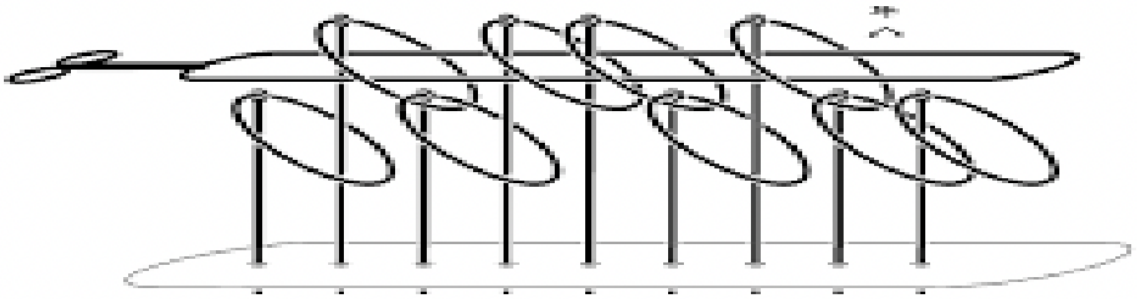

Figure 1.6 Configuration of Rings in the Chinese Ring Puzzle

As seen from Figure 1.6, the rings can either be in two states: on or off. While it may initially seem like simple manipulation can solve this puzzle, this is in fact false and this challenging puzzle follows a set of premises. Following these premises will allow for the puzzle to be solved through a tedious but methodological approach. The premises are summarized below.

1) Ring 1 can always be taken off or put on the handle
2) The only other ring that can be taken off or put on the handle is the ring immediately after the **lead** ring, which is defined as the first ring that is on the handle.
3) Premises 1 and 2 combine to summarize: At any point, there are exactly two rings that can be put on or taken off from the handle.

Therefore, given these premises, it is impossible to use a trial-and-error approach towards solving this puzzle, and instead, we can use the given premises to help solve the puzzle in a systematic manner. The method to remove 5 rings is outlined below and uses the application of the aforementioned premises.

1) Take off Ring 1 (Premise 1). Ring 2 becomes the lead ring.
2) Take off Ring 3 (Premise 2). Ring 4 becomes the lead ring.
3) Place Ring 1 on the handle (Premise 1). Ring 1 becomes the lead ring.
4) Take off Ring 2 (Premise 2).
5) Take off Ring 1 (Premise 1). Ring 4 becomes the lead ring.
6) Take off Ring 5 (Premise 2).
7) Place Ring 1 on the handle (Premise 1). Ring 1 becomes the lead ring.
8) Place Ring 2 on the handle (Premise 2).
9) Take off Ring 1 (Premise 1). Ring 2 becomes the lead ring.
10) Place Ring 3 on the handle (Premise 2). Ring 3 becomes the lead ring.
11) Place Ring 1 on the handle (Premise 2). Ring 1 becomes the lead ring.
12) Take off Ring 2 (Premise 2).
13) Take off Ring 1 (Premise 1). Ring 3 becomes the lead ring.
14) Take off Ring 4 (Premise 2).

15) Place Ring 1 on the handle (Premise 1). Ring 1 becomes the lead ring.
16) Place Ring 2 on the handle (Premise 2).
17) Take off Ring 1 (Premise 1). Ring 2 becomes the lead ring.
18) Take off Ring 3 (Premise 2).
19) Place Ring 1 on the handle (Premise 1). Ring 1 becomes the lead ring.
20) Take off Ring 2 (Premise 2).
21) Take off Ring 1 (Premise 1). Five rings are off

For simplification purposes, a binary solution where 0 represents a ring that is off and 1 represents a ring is on is presented below. The order of the rings is from left to right.

1) 01111
2) 01011
3) 11011
4) 10011
5) 00011
6) 00010
7) 10010
8) 11010
9) 01010
10) 01110
11) 11110
12) 10110
13) 00110
14) 00100
15) 10100
16) 11100
17) 01100
18) 01000
19) 11000
20) 10000
21) 00000

As seen, by following a combination of premise 1 and premise 2, we have devised a way of solving this impossible challenge for 5 rings in 21 steps, the minimum number of steps needed to remove the rings. It is here we begin to see mathematics play a role, and we can derive an equation to determine the minimum number of steps needed to remove n number of rings in the Chinese Ring Puzzle.

Mathematical Derivation

As mentioned in the above center, the solution to the nine-ring puzzle requires a methodological approach where mathematics can be integrated.

Advanced Topics Chinese Ring Puzzle (5 Rings) Project

The first area where mathematics plays a role in the puzzle is in the minimum number of "moves" needed to remove n number of rings. In order to derive this, we examine the minimum number of moves needed to solve a different number of rings, which are summarized in the table below. For the purpose of this derivation, the number of moves was determined by solving the puzzle manually for the given number of rings.

Number of Rings	Moves
1	1
2	2
3	5
4	10
5	21
6	42
7	85
8	170
9	341

Figure 2.1: Minimum number of moves needed to remove n rings from the puzzle

From observing the values in the table, it can be determined that a piecewise function, split between odd and even values of n, will need to be utilized to derive a function to model the minimum number of moves needed.

For an odd value of n, the number of moves increases in the following manner:

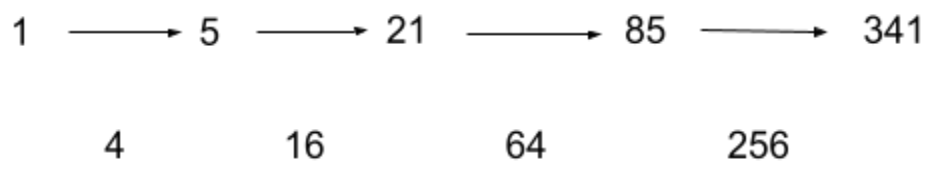

Figure 2.2: Representation for odd values of n

As seen, this resembles a quadratic sequence (not exactly) where we see the 2nd difference are all powers of four. Therefore, using this information, we can derive the following equation to represent the minimum number of values for odd values of n.

$$\frac{1}{3}(2^{n+1} - 1)$$

Equation 2.1: Minimum number of moves needed to solve n number of rings (Odd)

The same reasoning can then be applied to solve for even values of n.

For an even value of n, the number of moves increases in the following manner:

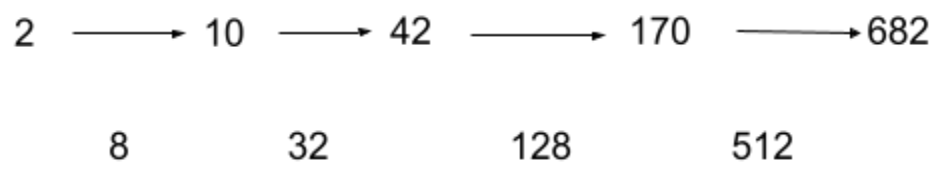

Figure 2.3: Representation for even values of n

Again, this has a close resemblance to a quadratic sequence, and again the 2nd difference follows a geometric series where the common difference is X4. We can again use the information and equation 2.1 to derive the following equation to represent the minimum number of values for even vales of n.

$$\frac{1}{3}(2^{n+1} - 2)$$

Equation 2.2: Minimum number of moves needed to solve n number of rings (Even)

Combining, the two equations together, we develop the following piecewise function to represent the minimum number of moves needed to solve the nine ring puzzle for any given value of n.

$$a(n) = [\tfrac{2}{3}(2^n - 1)] = \begin{cases} \frac{1}{3}(2^{n+1} - 2) & n \text{ even} \\ \frac{1}{3}(2^{n+1} - 1) & n \text{ odd}, \end{cases} \qquad (1)$$

Equation 2.3: Combined Equation for the number of moves needed to solve the CRP.

This equation that was derived is closely related to the concept of gray codes, which we decided to investigate through our literature review of a 2017 paper by members of Stanford University and Morrison Academy.

Representation of Gray Codes

Advanced Topics Chinese Ring Puzzle (5 Rings) Project

The simplest way to represent the rings numerically, as shown previously, is through 0's and 1's, with the former representing a ring that is "off" and the latter a ring that is "on". In terms of programming, this may immediately suggest a base two approach. However, this actually isn't the most efficient, since consecutive values in base ten values differ by more than one bit.In other words, in binary, the numbers increase in the following format: 00→01→10→11. However, this does not mirror the movement of the rings, which can only be removed when all rings except for the one directly next to it have been removed. Gray Code creates the exact representation we need: 00→01→11→10. Using Gray Codes, every "movement" of the rings results in a one-bit shift, which perfectly mirrors the movement of the rings.

```python
# Function called upon by the Ring Solving function to iterate the current ring state
def greySteps(i):
    if (i == 1): return ["0", "1"]
    previous = greySteps(i - 1)
    updated, new = [], []
    for step in previous:
        updated.append("0" + step)
        new.append("1" + step)
    return updated + new[::-1]

# Recursive function to solve the rings
def solveRings(i):
    steps = greySteps(i)[::-1]
    for j in range(len(steps)):
        if steps[j] == "1"*i:
            return(steps[j:])

# Uses the piecewise function we found through our research to count the minimum number of steps
def minSteps(i):
    if i % 2 == 0:
        return(1/3*(2**(i+1)-2))
    else:
        return(1/3*(2**(i+1)-1))
```

Fig 3.1: The two main functions of the program, and the final function to count the minimum number of steps

Our program consists of two main functions: one that we call to solve an *n* ring puzzle, and one that the first function calls to iterate the current ring state recursively. Let the former be called *solveRings* and the latter *greySteps*. Let us consider a three ring puzzle, with the starting configuration of ['111']. We first call *solveRings* with an argument of 3. This calls *greySteps* with an argument of 3. The code then iterates the argument by -1 until it reaches an argument of 1, creating a recursive stack. At this point, it imitates the configuration of bits in Gray Code, returning values of ["00", "01", "11", "10"]. This value is then passed into *greySteps(3)*, which is at the top of the stack. After passing the previously returned values through the for loop again, the output ultimately returned by *greySteps(3)* is ["000", "001", "011", "010", "110", "111", "101", "100"].

Now that the program has exited the recursive stack created by *greySteps*, it returns to *solveRings*. The value returned by *greySteps(3)* above is assigned to the variable *steps*. It then

loops through *steps*, removing any configurations that appeared prior to '111', our initial configuration, in the sequence. Finally, the values left are returned by the program as the sequence of steps required to solve a three ring puzzle.

The program also uses the piecewise function determined earlier in this paper to count the minimum number of steps taken to solve the puzzle. Testing this program has demonstrated that the chosen algorithm does indeed use exactly the minimum number of steps. In the case of the three ring puzzle, this is ['111', '110', '010', '011', '001', '000'], or five steps, excluding the initial state.

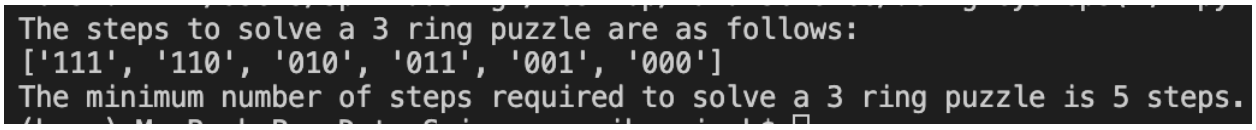

Fig 3.2: The output of our program for 3 rings, with 5 steps

This method can be applied to any number of rings. However, the recursive stack created by *greySteps* becomes quite complicated for large values, making it far more difficult to follow for our initial nine-ring problem. To see a visual representation of each step of the program, visit this link to see the 3-ring example.

Fig 3.3: The output of our program for 9 rings, with 341 steps

Advanced Topics Chinese Ring Puzzle (5 Rings) Project

```
ringCount = 5
print("The steps to solve a "+ str(ringCount) + " ring puzzle are as follows:")
print(solveRings(ringCount))
print("The minimum number of steps required to solve a " + str(ringCount) + " ring puzzle is "+ str(int(minSteps(ringCount))) + " steps.")

ringCount = 9
print("The steps to solve a "+ str(ringCount) + " ring puzzle are as follows:")
print(solveRings(ringCount))
print("The minimum number of steps required to solve a " + str(ringCount) + " ring puzzle is "+ str(int(minSteps(ringCount))) + " steps.")
```

Fig 3.4:The function call can be personalized to any number of rings by changing the value of ringCount