Hanoi towers

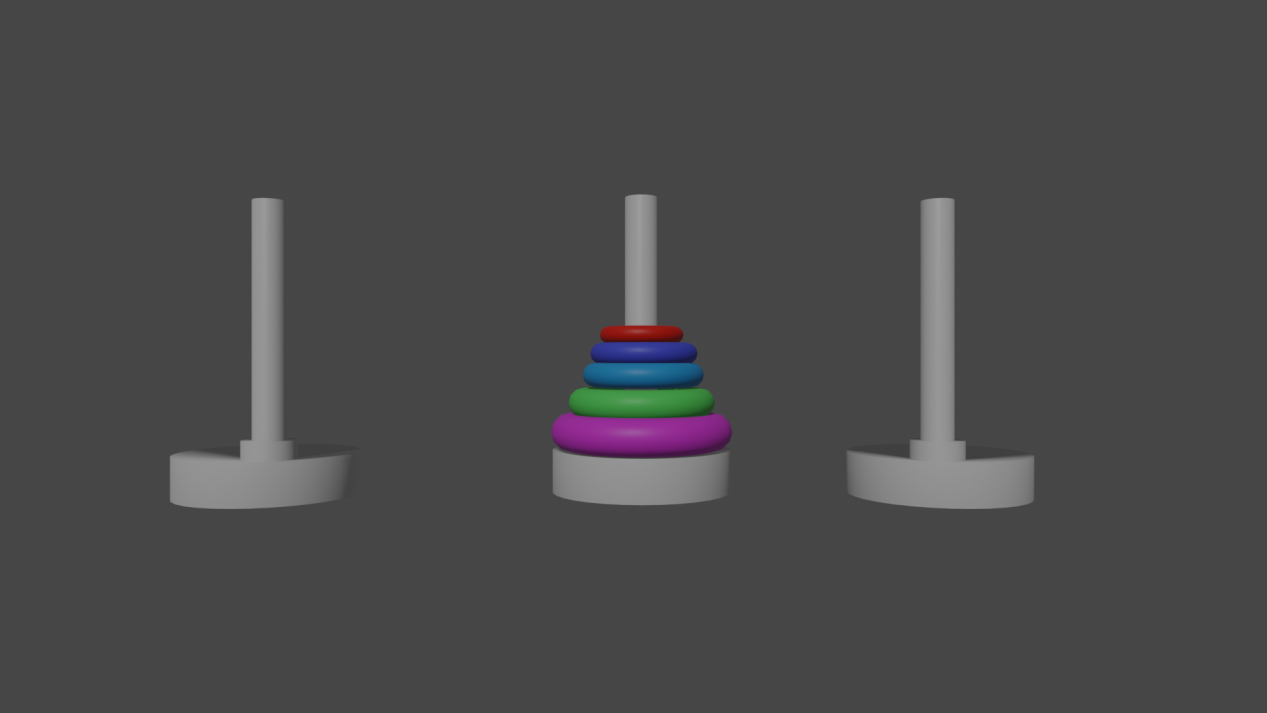
Three pegs n disks (initially placed one on top of the other in a growing order from top to bottom)

We need to move the stack from one peg to an other with only two rules

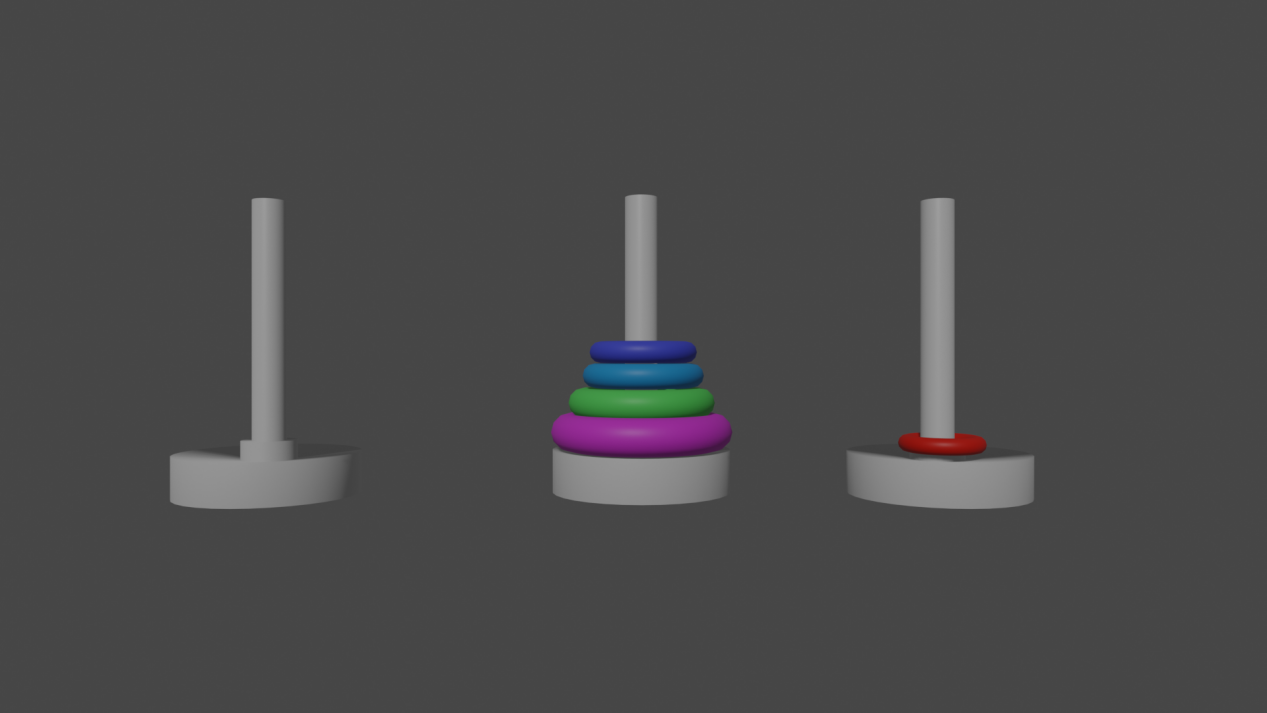
+only move one disk at a time

+no disk can be placed on top of on that is smaller

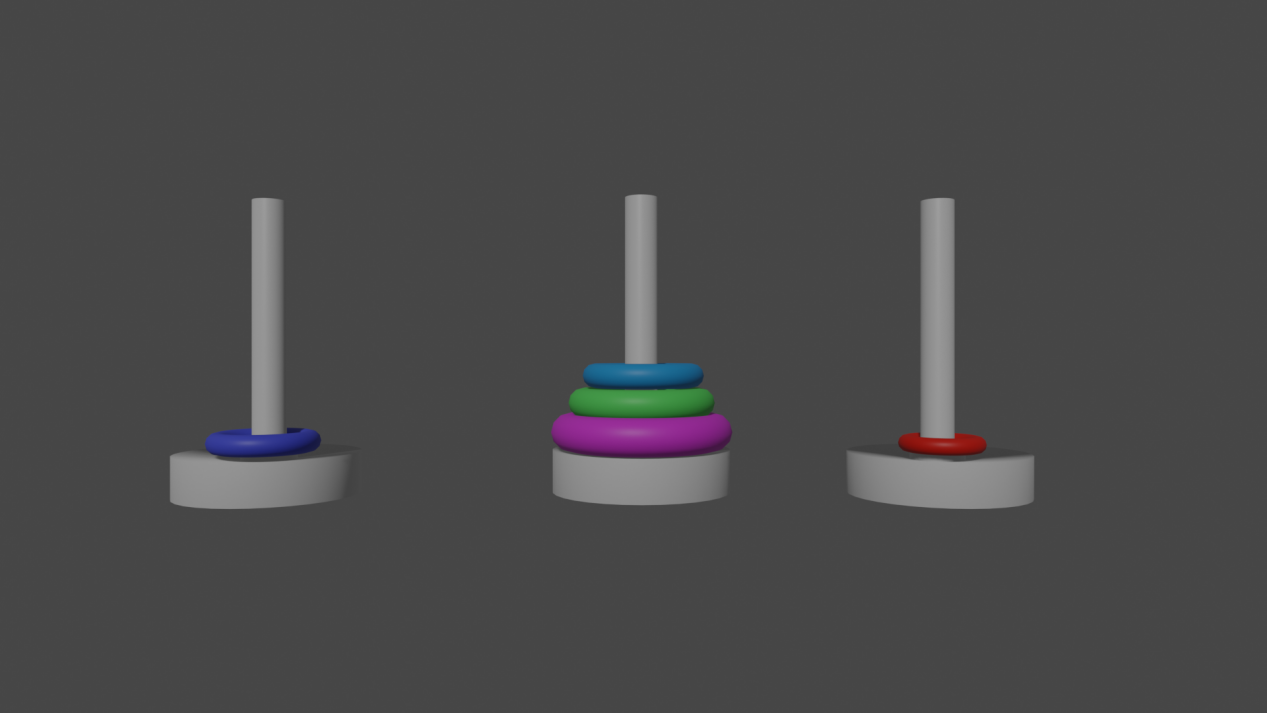
A)



B)



C)



And so on

Haven’t read the solution but i know of the existence of a classic recursive solution

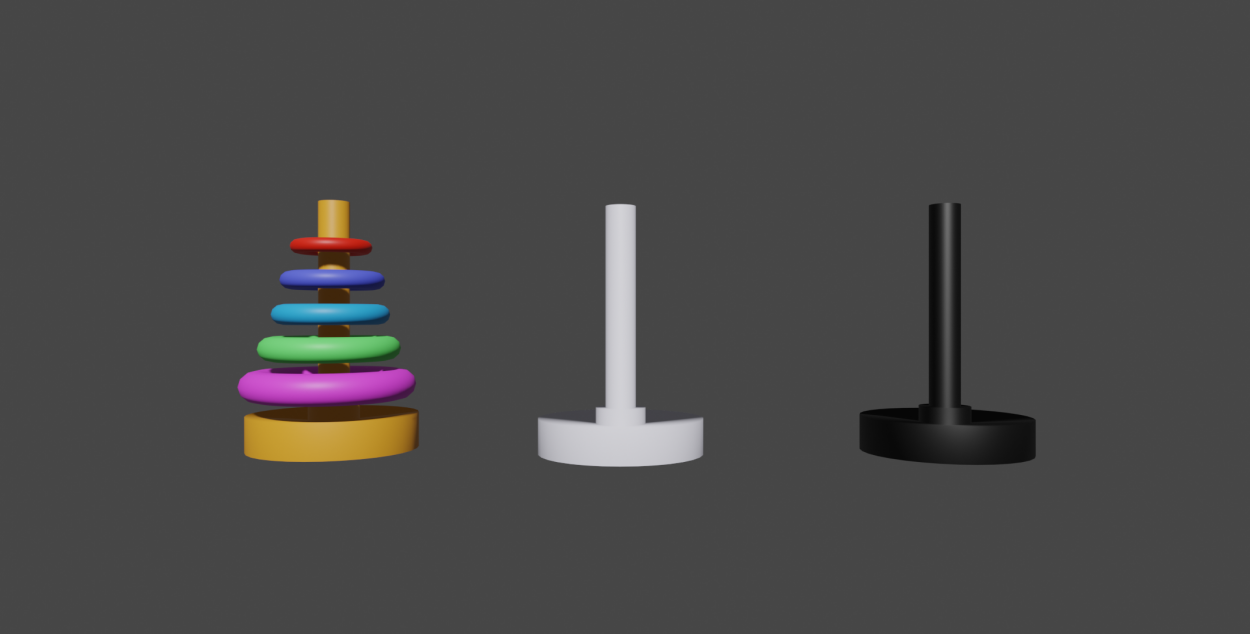
To wrap my head around the solution i resorted to a small model in blender

I will write the code in python:

An idea that occurred to me is to be able to solve this every time i evaluate the top of the pegs and try always to move the biggest loop when possible towards the smallest possible path without going back

Is it a brut force algorithm ?

One rule that might be helpful is once u move a disk into a peg it would be unreasonable to move that disk again



Wherever i place the top red disk the second one after it (turquoise) will be in the same peg

Same thing for the purple one

So if n = 2k+1 i need to put the top most disk (red ) on the target peg (white for example)

After further observation i can see two phases

Phase one

i put the top peg on the most filled non target peg

The following are two runs with 2n and 2n+1

To be a bit more organized i’m gonna need to add more rules

After trial and error i could observe that

+only move one disk at a time

+no disk can be placed on top of on that is smaller

+can’t move a disk twice

+if 2k under disk move it to target

+if 2k+1 under disk move it to non target

+if disk on target target becomes for that disk the non init peg

Testing the rules above

They work but unfortunately this method is more of an iterative code, even though a little tricky, but it can be turned into a recursive one

The iterative solution i found in hacker earth goes as follows

U make a function and enter as parameters the number moves required to solve it

[Data Structure & Algorithms - Tower of Hanoi (tutorialspoint.com)](https://www.tutorialspoint.com/data_structures_algorithms/tower_of_hanoi.htm)

(ctrl+left click it)

The idea is

To divide the stack of n disks into two:

+The bottom one

+And the rest n-1

you then move the n-1 disks to the auxiliary peg

move the bottom to the destination

Then move those n-1 disks to the destination

STARTProcedure Hanoi(disk, source, dest, aux)

IF disk == 1, THEN

move disk from source to dest

ELSE

Hanoi(disk - 1, source, aux, dest) // Step 1

move disk from source to dest // Step 2

Hanoi(disk - 1, aux, dest, source) // Step 3

END IF

END Procedure

STOP

Hanoi(4,source,dest,aux)=>hanoi(3,source,aux,dest) disk to dest hanoi(3,aux,dest,source)

Hanoi(3,source,aux,dest)=>hanoi(2,source,dest,aux) disk to aux hanoi(2,dest,aux,source)

Hanoi(3,aux,dest,source)=>hanoi(2,aux,source,dest) disk to dest hanoi(2,source,dest,aux)

hanoi(2,source,dest,aux)=>hanoi(1,source,aux,dest) disk to dest hanoi(1,aux,dest,source)

hanoi(2,dest,aux,source)=>hanoi(1,dest,source,aux) disk to aux hanoi(1,source,aux,dest)

hanoi(2,aux,source,dest)=>hanoi(1,aux,dest,source) disk to source hanoi(1,dest,source,aux)

hanoi(2,source,dest,aux)--same as the top one above (a problem of dynamic programming )

hanoi(1,source,aux,dest) =>move disk from source to aux

hanoi(1,aux,dest,source)=>move disk from aux to source

hanoi(1,dest,source,aux) =>move disk from dest to aux

hanoi(1,source,aux,dest)=>...

For 10 disks there were 1023 move needed 2^10 -1 =1024-1=1023 so the count is right

For 4 disks 15 moves were needed so the count is right again