Intro to Coding with Python– Code Reuse

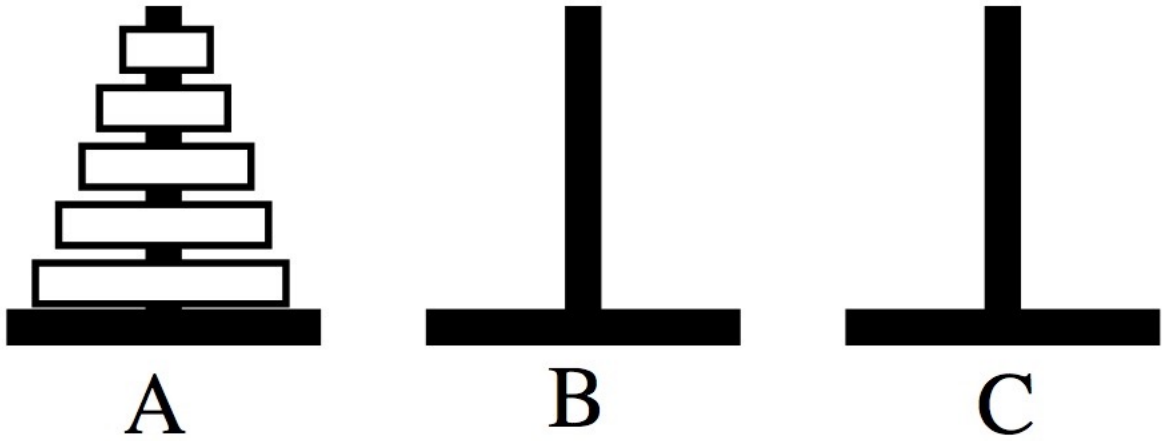
Dr. Ab Mosca (they/them)

Slides based off slides courtesy of Jordan Crouser (https://jcrouser.github.io/)

Plan for Today

- Motivating example: Towers of Hanoi
- Tough problems, simple solutions

Towers of Hanoi



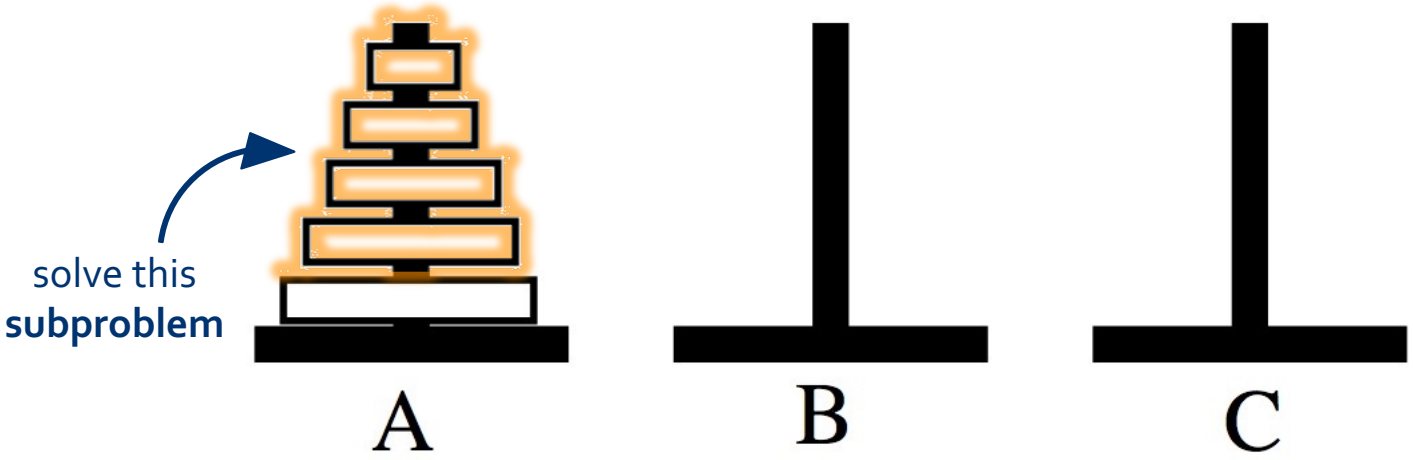
Rules of the game

- You can only move **one** disk at a time
- You can only move a disk to a pole where it will be the **smallest** (i.e. you can't put a disk on top of a larger one)
- You can only remove the **smallest** disk from a pole (i.e. you can't lift up the stack to get a larger disk from below)

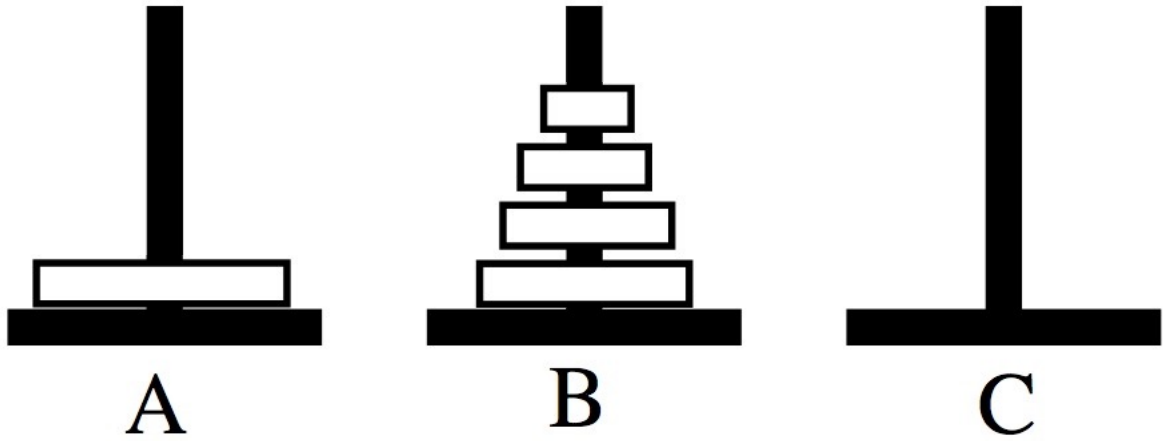
Discussion

Notice any **patterns**?

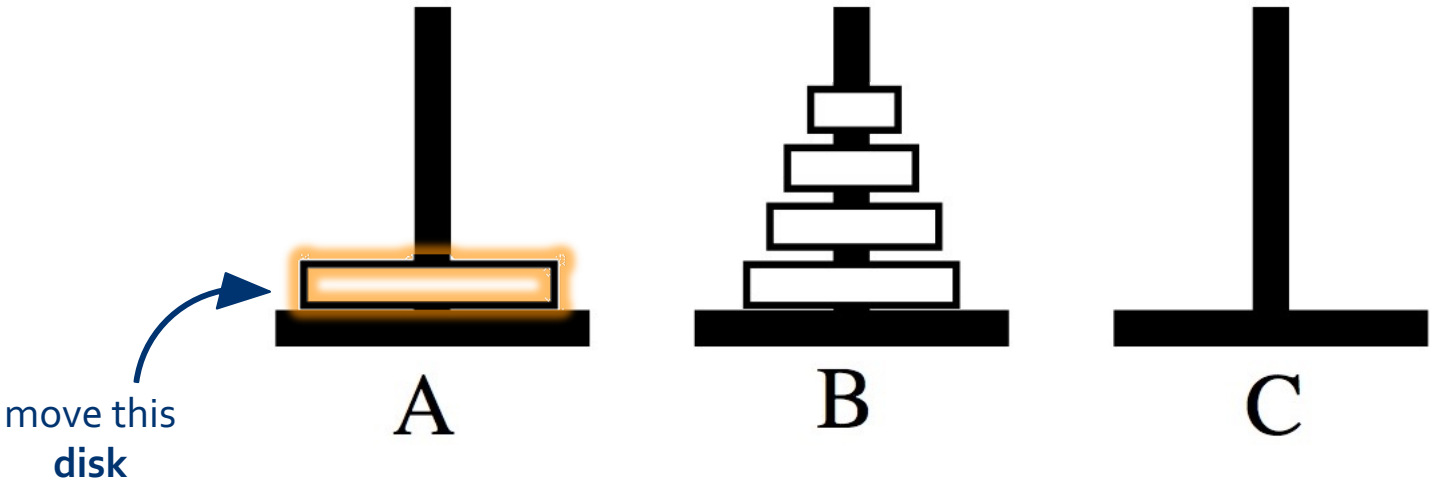
Recursive Towers



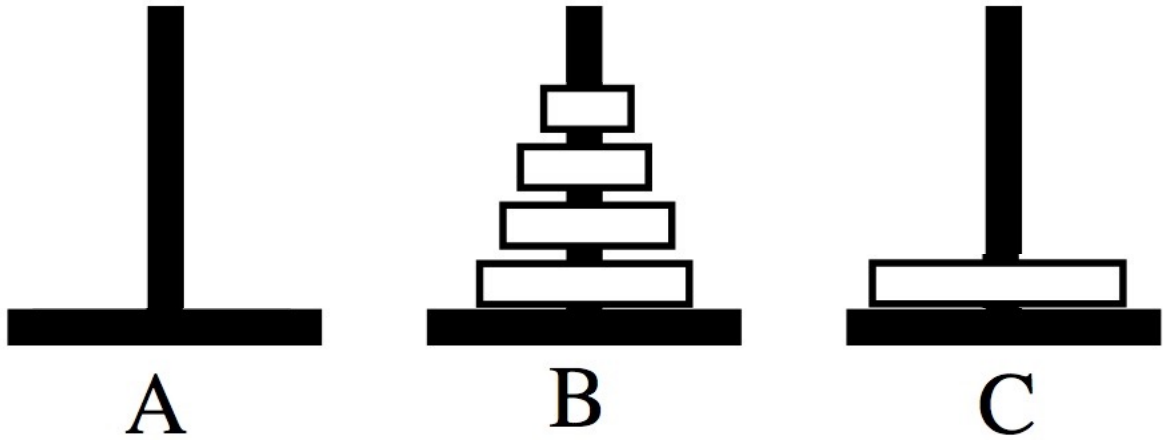
Recursive Towers



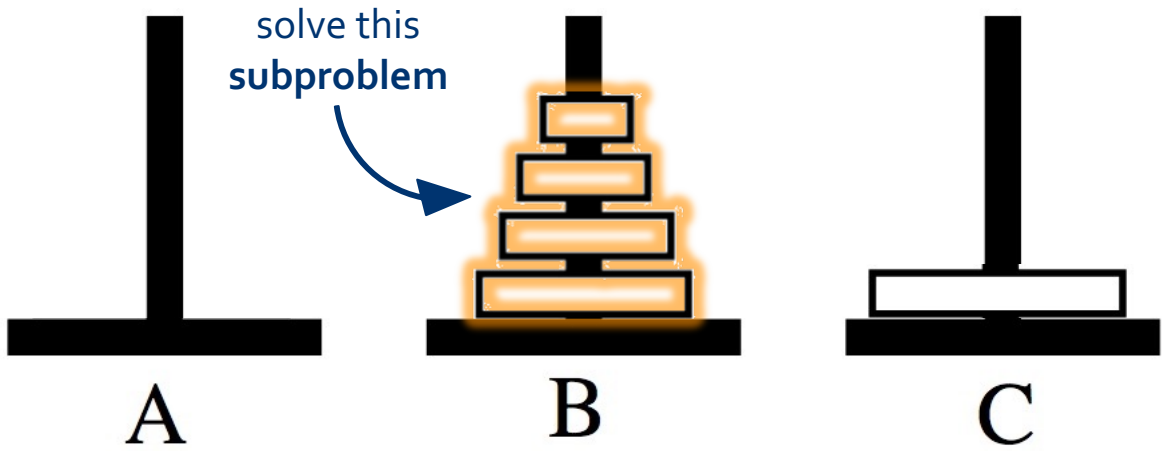
Recursive Towers



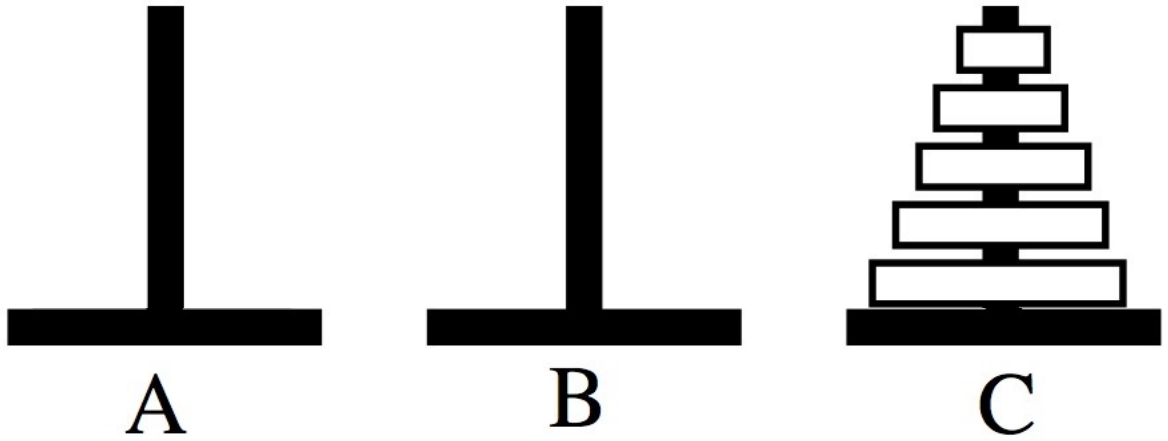
Recursive Towers



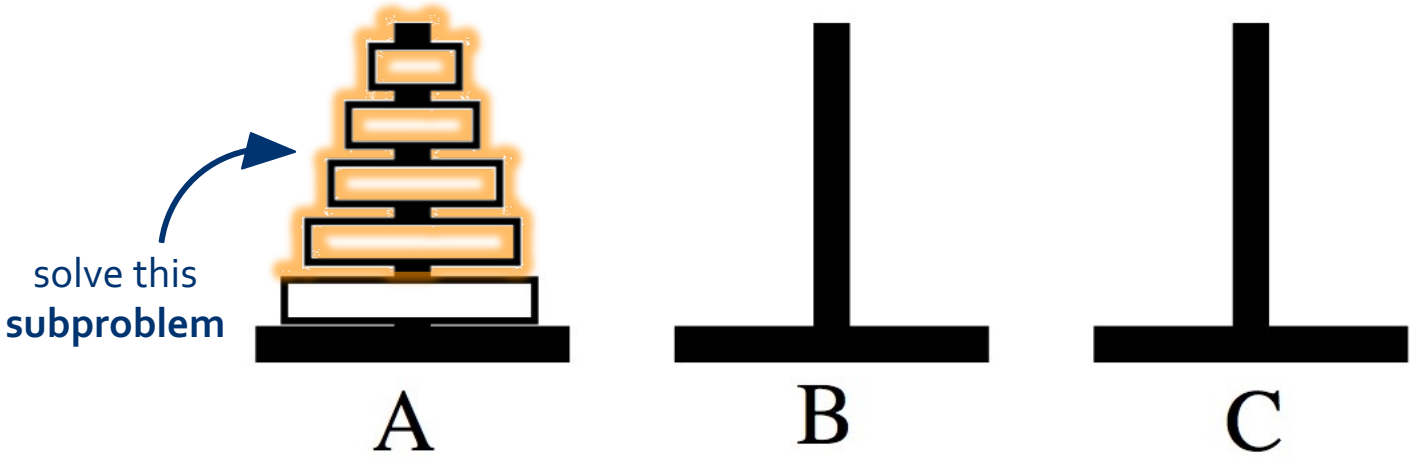
Recursive Towers



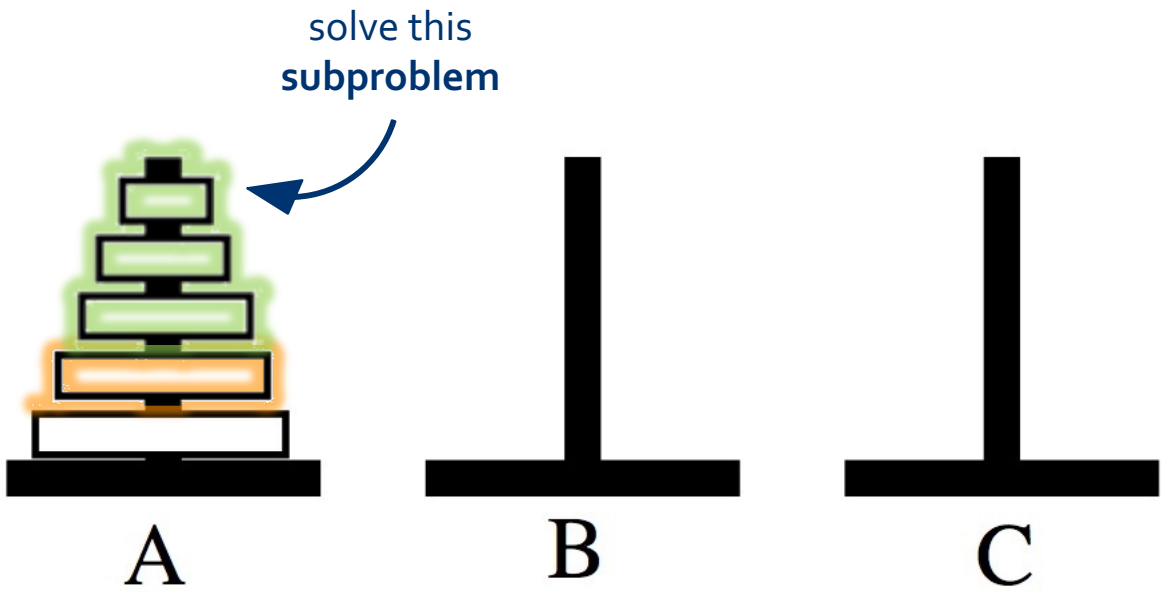
Recursive Towers



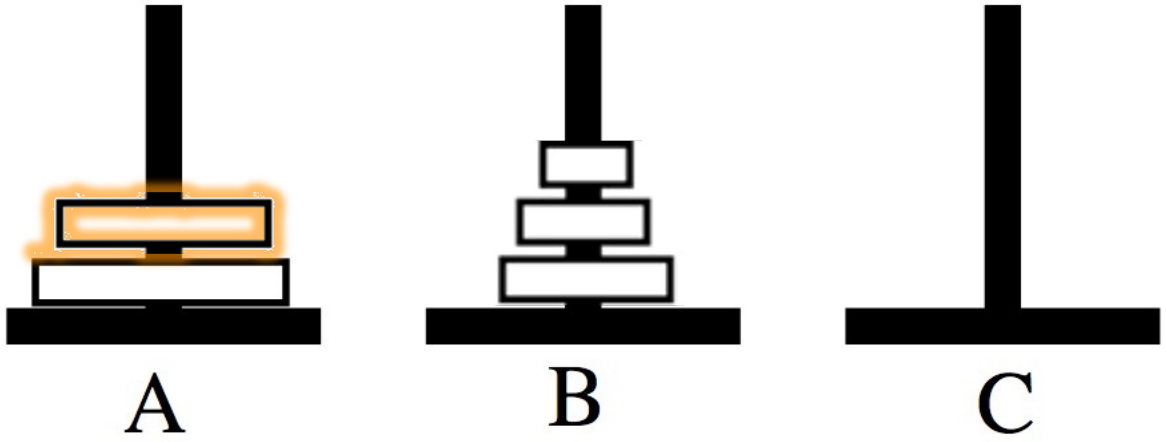
Recursive Towers



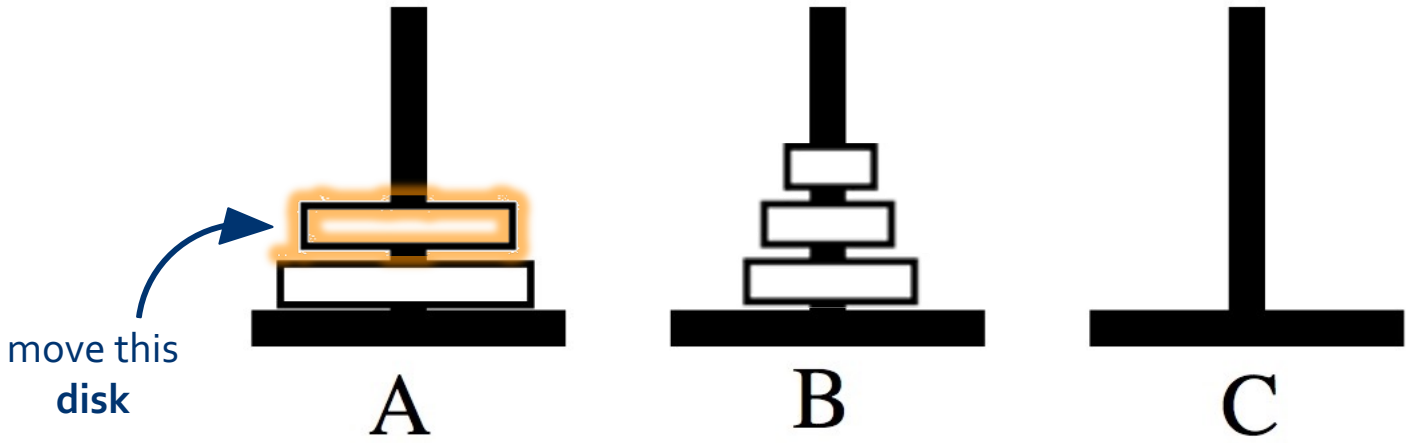
Recursive Towers



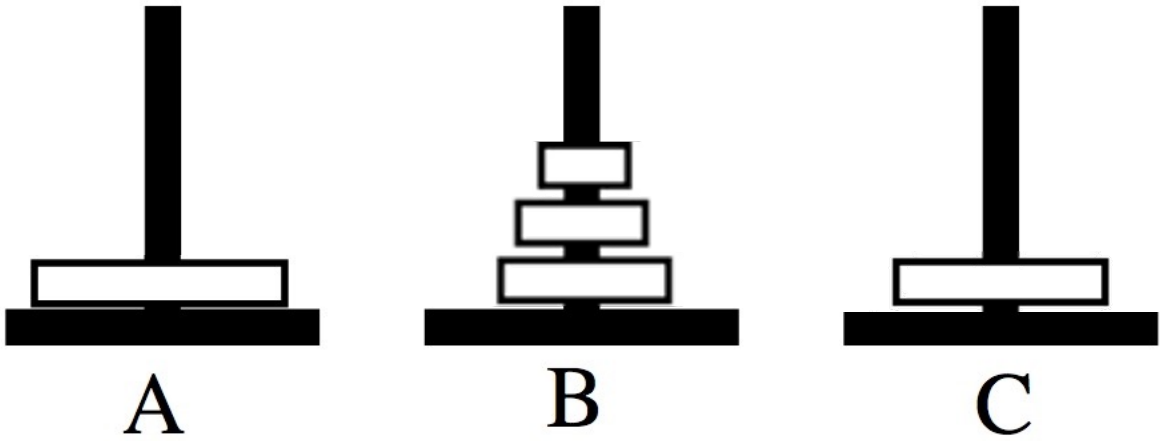
Recursive Towers



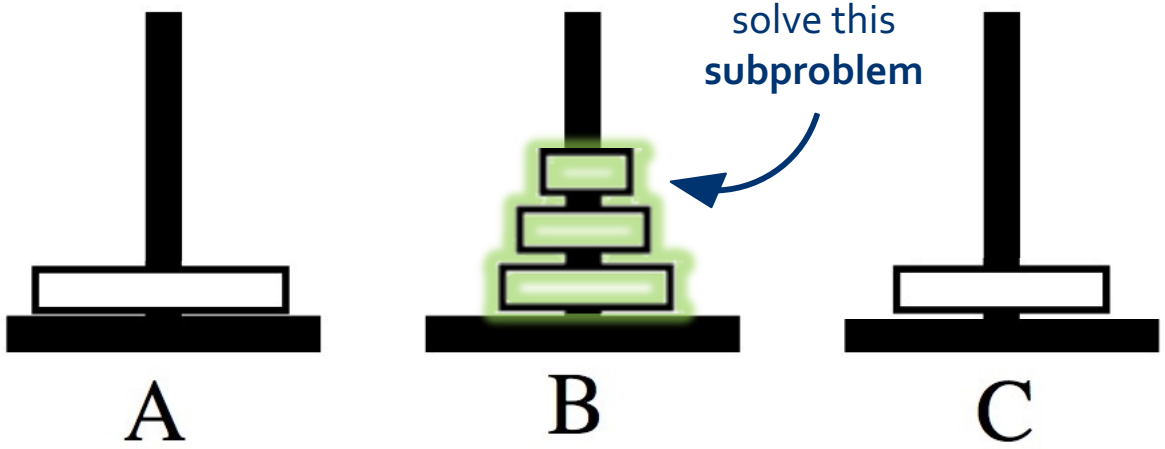
Recursive Towers



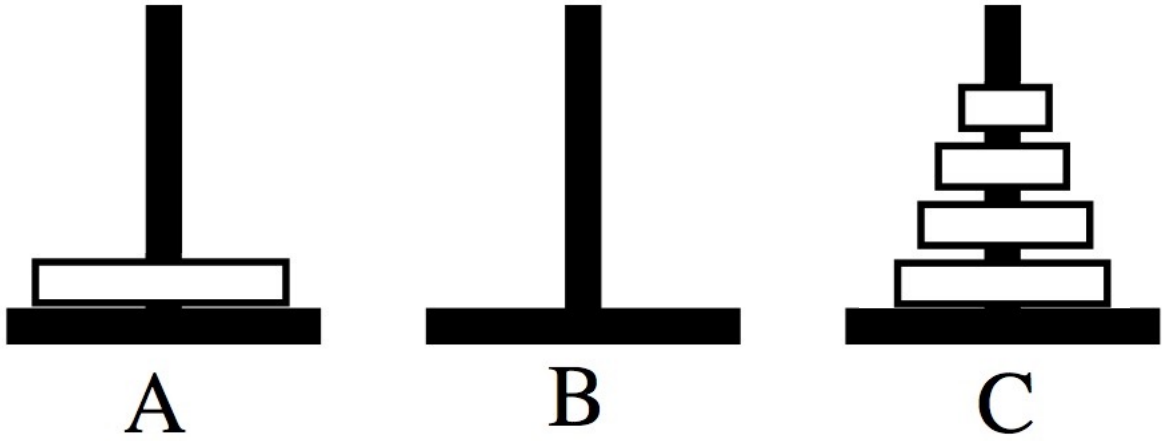
Recursive Towers



Recursive Towers



Recursive Towers



...and so on!

Discussion

How many **moves** does it take?

Algorithmic analysis

nDisks	nMoves
1	1
2	3
3	7
4	15
5	31
6	64
7	127

Notice any **patterns**?

$$nMoves = 2^{nDisks} - 1$$

Basic structure of a recursive algorithm

- **A base case**: what to do in the simplest possible case (i.e. when you have a single disk)
- **A recursive step**: break the original problem into one or more smaller problems, and solve that (saving the intermediate result)

Demo: Towers of Hanoi in Python