CSCA48 Winter 2018
Week 9:Recursion

Marzieh Ahmadzadeh, Nick Cheng

University of Toronto Scarborough

Trees: recursion definition

- **Tree**:
  - either an empty pointer (base case : `p is None`)
  - or is made of a single node (`root`), where its children are a **tree** (recursive case)

- **Binary Tree**:
  - is either empty (base case -> `pointer is None`)
  - or is made of a single node (`root`), where the left and right children are a **binary tree** (recursive case)

Trees: recursion definition

- **Depth** of a node:
  - If p is the root then depth(p) is 0.
  - Otherwise 1+**depth**(p's parent)

- **Height** of a tree:
  - If p is a leaf then height(p) is 0
  - Otherwise 1 + max (**height** of its children)

Trees: recursion definition

- **pre-order** traversal:
  - visit the root
  - **pre-order**(left subtree)
  - **pre-order**(right subtree)
- **In-order** traversal:
  - **In-order**(left subtree)
  - visit the root
  - **In-order**(right subtree)
- **post-order** traversal:
  - **post-order**(left subtree)
  - **post-order**(right subtree)
  - visit the root

Binary Search

- We can implement an effective searching algorithm, without a need to create a BST.
- This algorithm is applicable to a sorted list.
- Recursively:
  - Find the middle of the list
  - If what you are looking for is greater than the item in the middle, search the right sub-list
  - Otherwise search the left sub-list
  - Stop either when you find the data or when the lower_boundary_index > higher_boundary_index

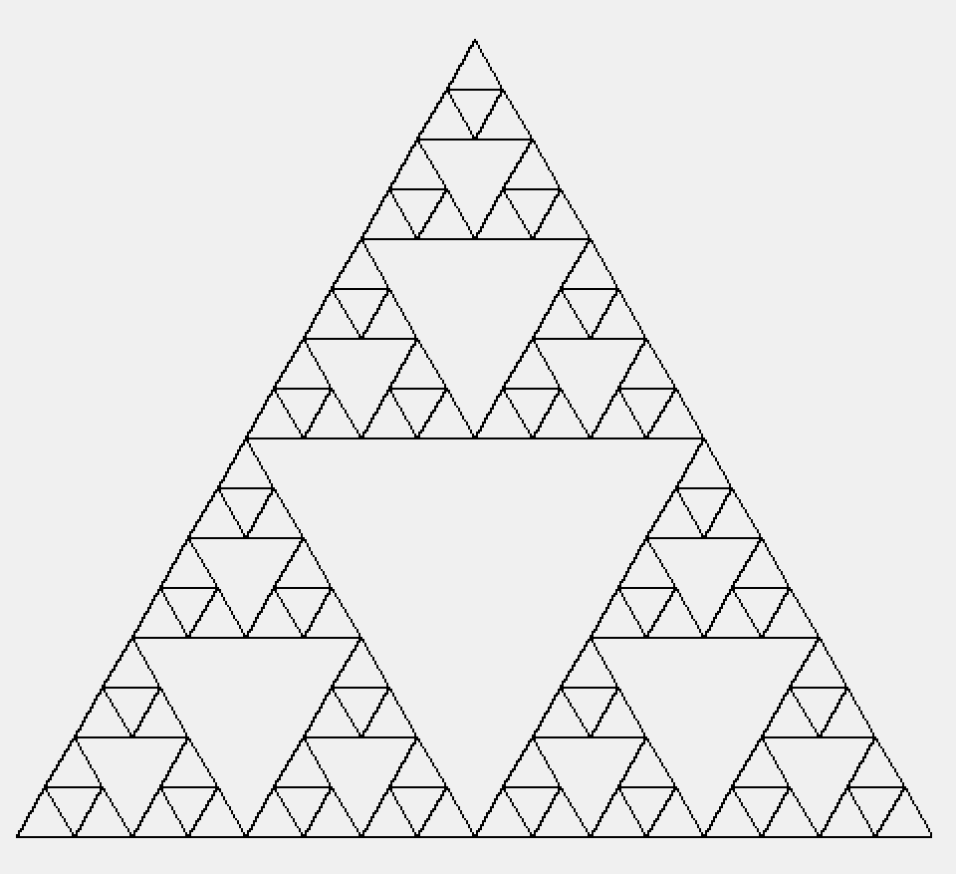
Binary Search Tree

- Recursive implementation of a BST
  - Search
  - Find
  - Insert
- Tree rotation

Towers of Hanoi

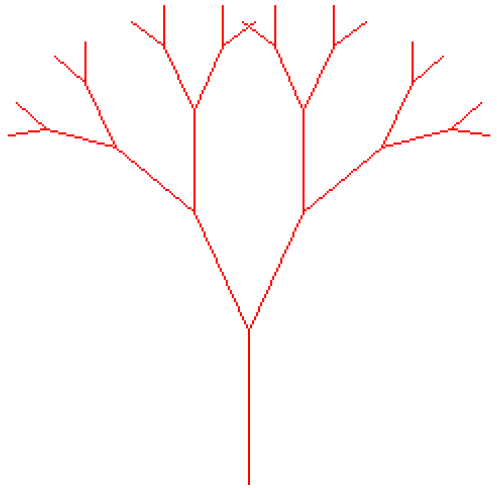
- Goal: generate an instruction that says how to move all the discs from one pole to another providing that:
  - Only one disc can be moved at a time.
  - A disc can be placed either on empty pole or on top of a larger disc.
- How many moves are required to move all n disks?
  - $2^n - 1$

Sierpinski Triangle



Fractal Tree

- A tree that looks the same no matter how much you magnify it.



Final Words

- Solve the problem with a small input size first
- Gradually add to the input and find the recursive solution
- Have both direct and indirect recursion in mind