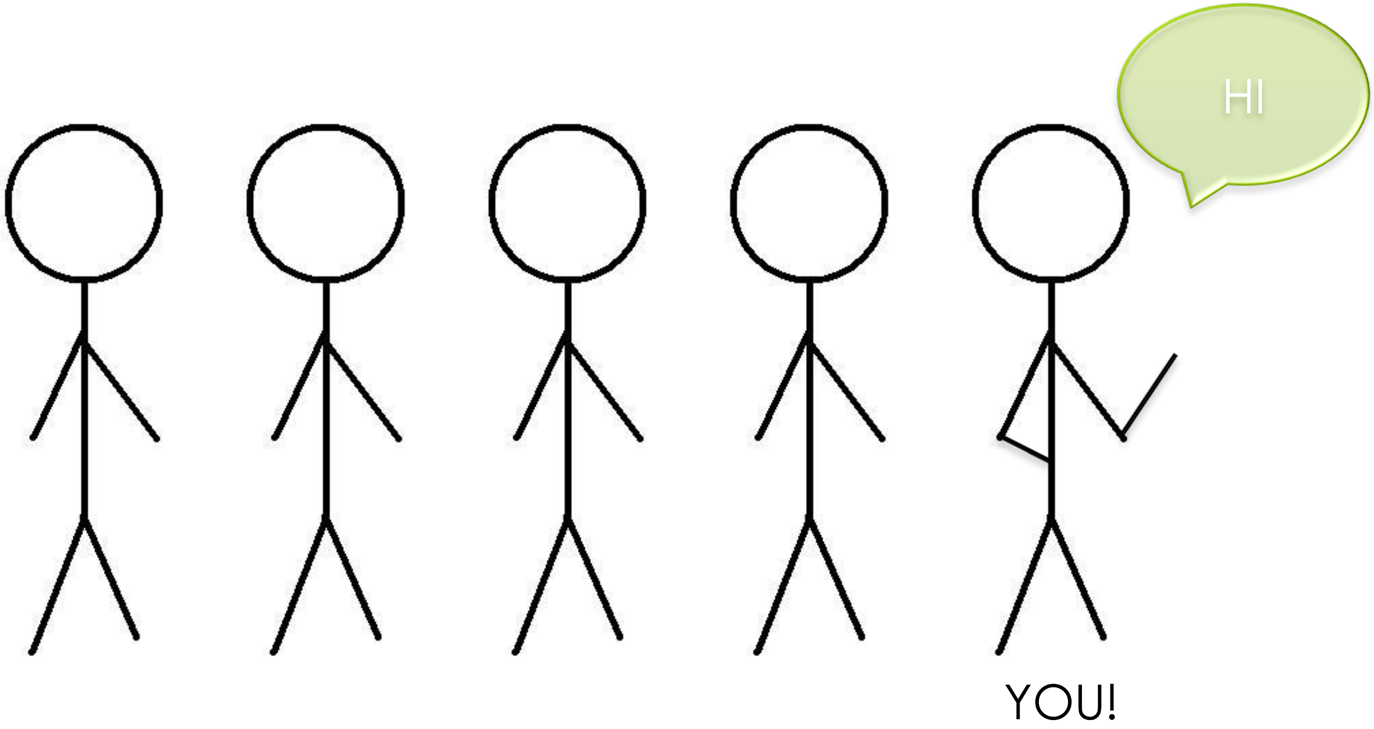
Recursion

Suppose You Are Waiting in Line…



… A Very Long Line

You want to determine how many people are in front of you, but you **cannot see** and you're **not allowed to move**. You are **only** allowed to **speak to the person in front of you and behind you**. How do you do it?

Recursion

- A programming technique that breaks down a complex problem into smaller, manageable pieces
- Recursive solutions solve a problem by applying the same algorithm to each piece and then combining the results.

The General Formula

#people in front of person in front of you +

person in front of you =

# people in front of you

The Solution

1. Tap the shoulder of the person in front of you and ask how many people are in front of him/her

2. Wait for his/her response and add 1

1. If someone asked, tell them how many people are in front of you

A Diagram

- Ask and wait
  - Ask and wait
    - Ask and wait….
      - …. Reached first in line. Tell person behind it is 0.
    - Tell person behind it is 0+1
  - Tell person behind it is 1+1…
- Tell Person behind it is x +1

Recursive Algorithms

- There are two main components to recursive algorithms

  - 1) Base Case

  - 2) The Recursive Case

Recursive Algorithms

- There are two main components to recursive algorithms
  - 1) Base Case: The point where you stop applying the recursive case
  - 2) Recursive Case: The set of instructions that will be used over and over

In the Queue Problem…

- Recursive case is
  - Tap person in front of you. Ask #people in front of him. Wait for his answer and add 1.
- Base case is
  - person 0. You do not do execute the above.

Recursion and Programming

- A recursive function is a function that calls itself

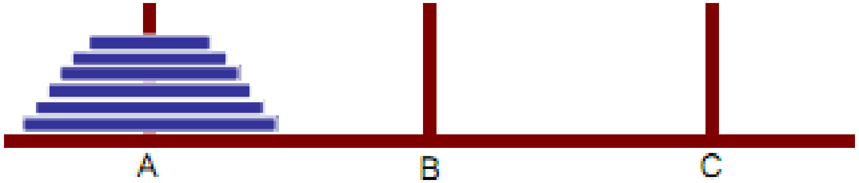
```
numberOfPeopleInFront(person){
        If (there is no one to tap)
            return 0
        Else
           tap person in front of you (F)
           #ppl in front of F =  numberOfPeopleInFront(F)
           return #ppl in front of F + 1
}
```

Pseudo-code diagram

- Ask and wait
  - Ask and wait
    - Ask and wait….
      - …. Reached first in line. Return 0.
    - Return 0+1
  - Return 1+1…
- Return x +1

Towers of Hanoi

- A prominent recursive problem
- Starting Configuration:



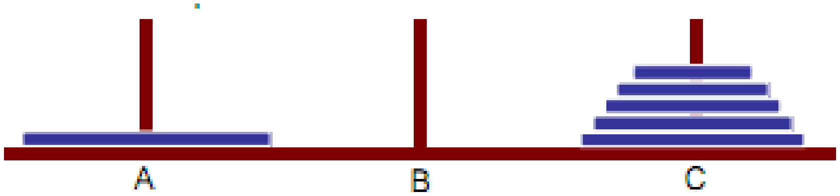
- Goal: Move tower from A to B

Rules

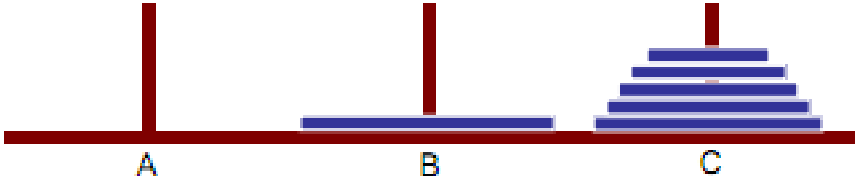
- Move one disk at a time
- A larger disk cannot be placed on top of a smaller disk
- We can use some needles as temporary storage

Subgoals

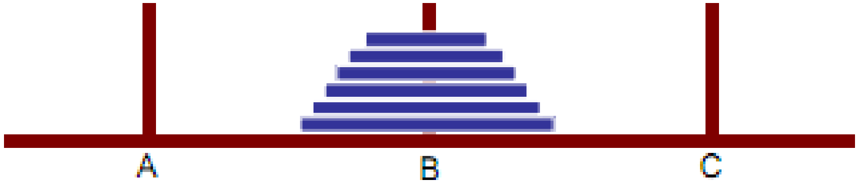
- Get top x-1 disks from A to C



- Get bottom disk from A to B



- Move top x-1 disks from C to B

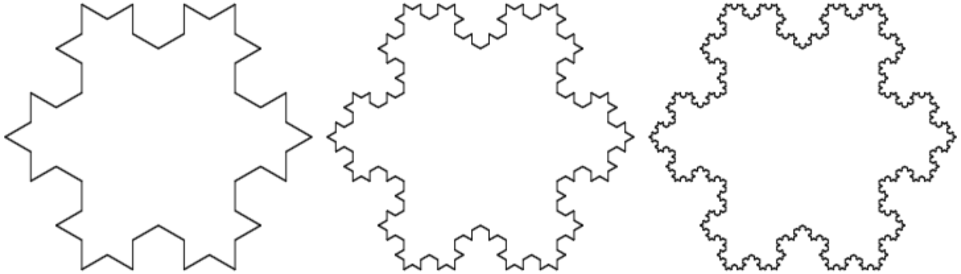


Recursion Behind Towers

- Base Case: Moving the Largest Disk to Needle B
- Recursive Case: Do same for the x – 1 disk above it
- http://www.mazeworks.com/hanoi/
- Fun Fact: It takes at least 2^n-1 moves to solve the puzzle

Fractals

- A rough or fragmented geometric shape that can be split into parts, each of which is (at least an approx. of) a reduced copy of the whole
- Base case: Starting shape
- Recursive case: Repeating shape in different sizes



Koch Snowflake