Dictionary

Creation

- x = { key:value, ..., key:value}

Access

- x[key]

Find the dictionary example here

- http://bit.ly/ZGuHwA

Lecture 6: Recursion

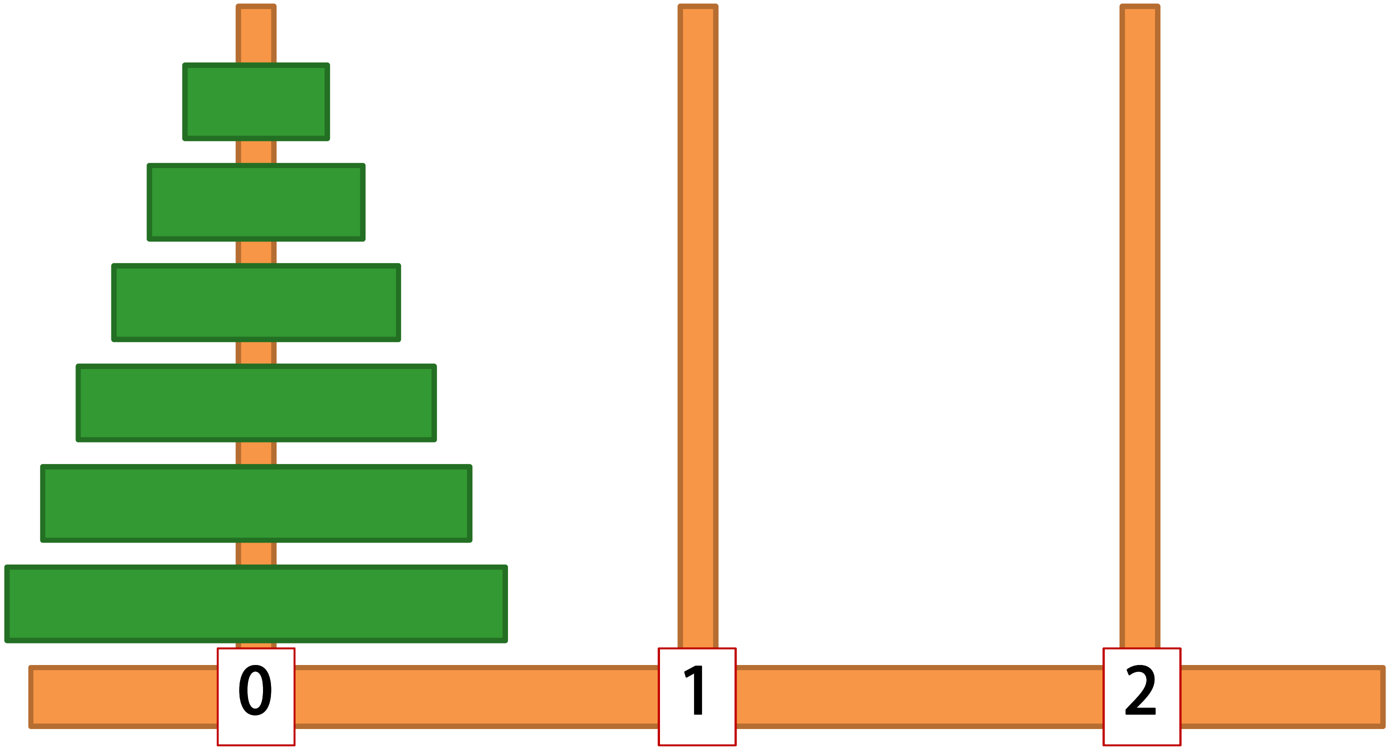
Armando Solar-Lezama

MIT COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE LABORATORY

Tower of Hanoi

Goal: Move all discs from source to destination

- You can never place a large disk over a smaller disk



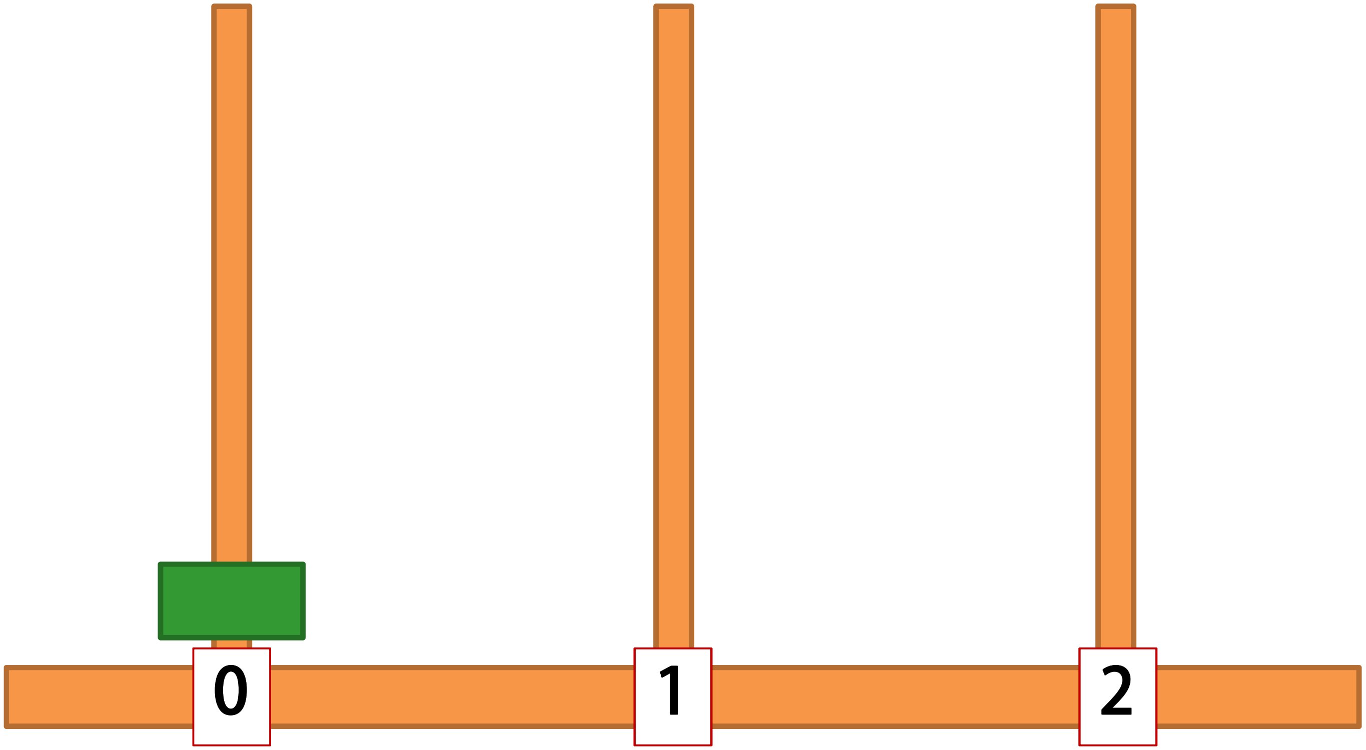
Recursion: key ideas

How do you know if recursion is right for you?

- Solving the problem in one shot is hard
- Solving a problem of size 1 is trivial
- If I give you a solution to a smaller problem, it’s easy to extend to a bigger problem

Solving a problem of size 1 is trivial

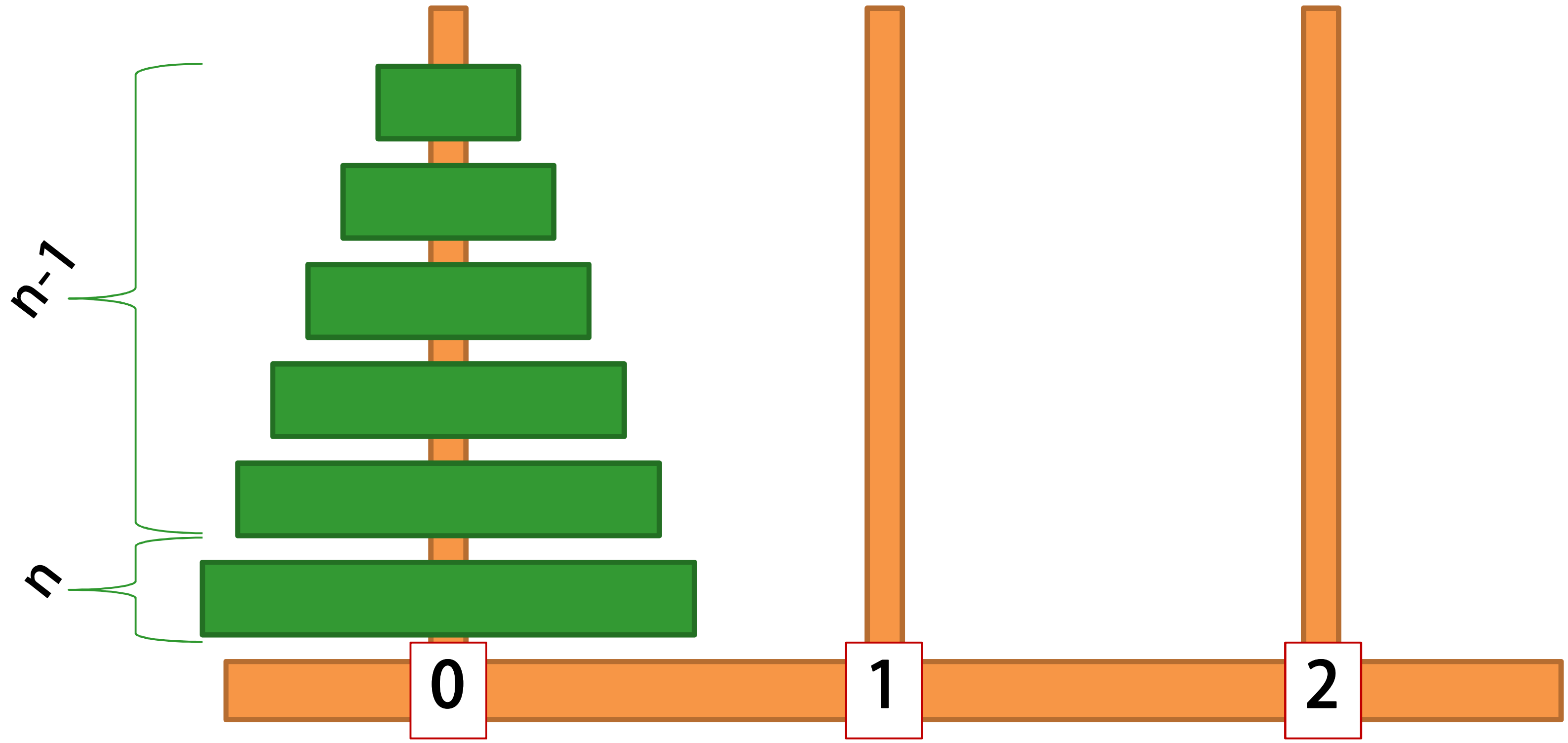
Could you solve the puzzle for 1 disk?



Extending from small to large

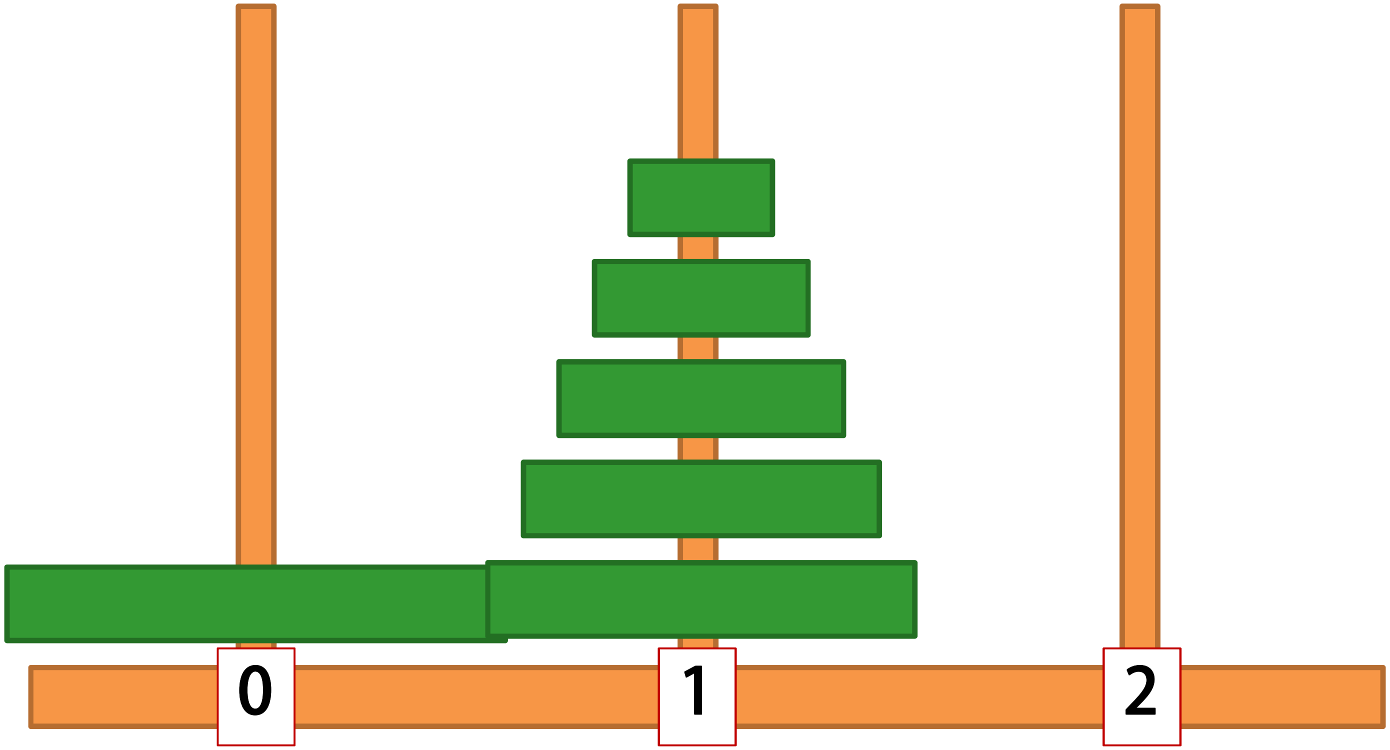
Suppose I know how to move n-1 disks following the rules

- How can I use this knowledge to move n disks?



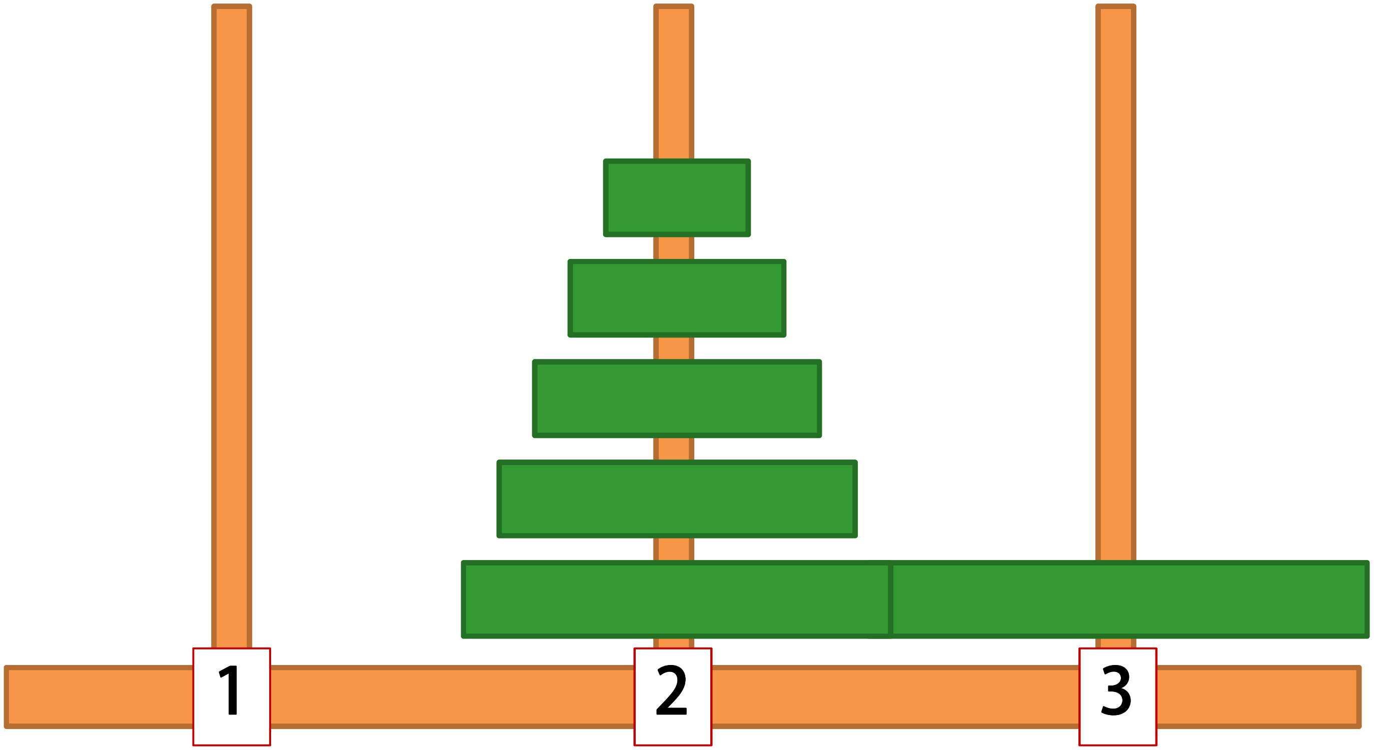
Extending from small to large

- Move n-1 to 2
- Move the nth disk to 3



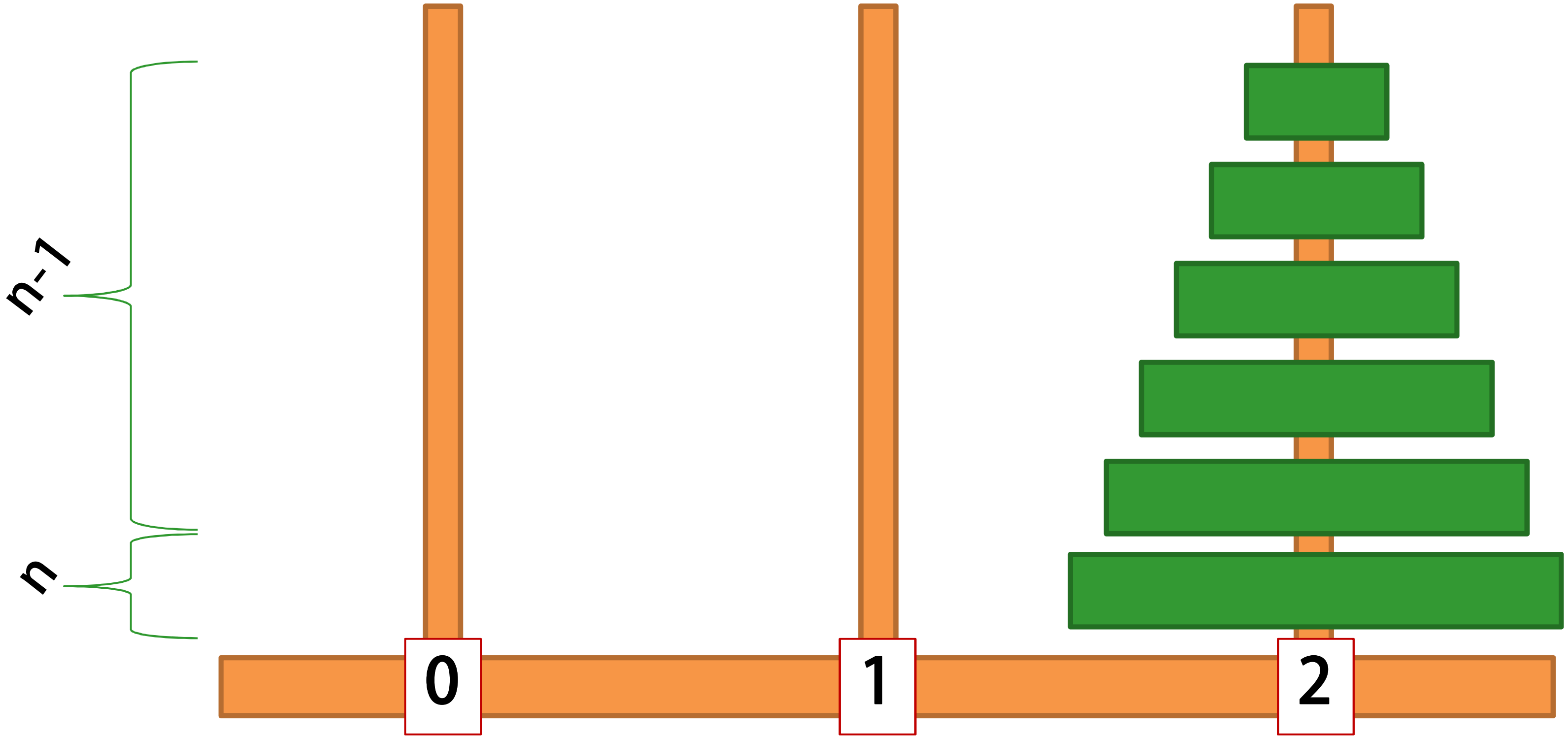
Extending from small to large

- Move n-1 to 2
- Move the nth disk to 3

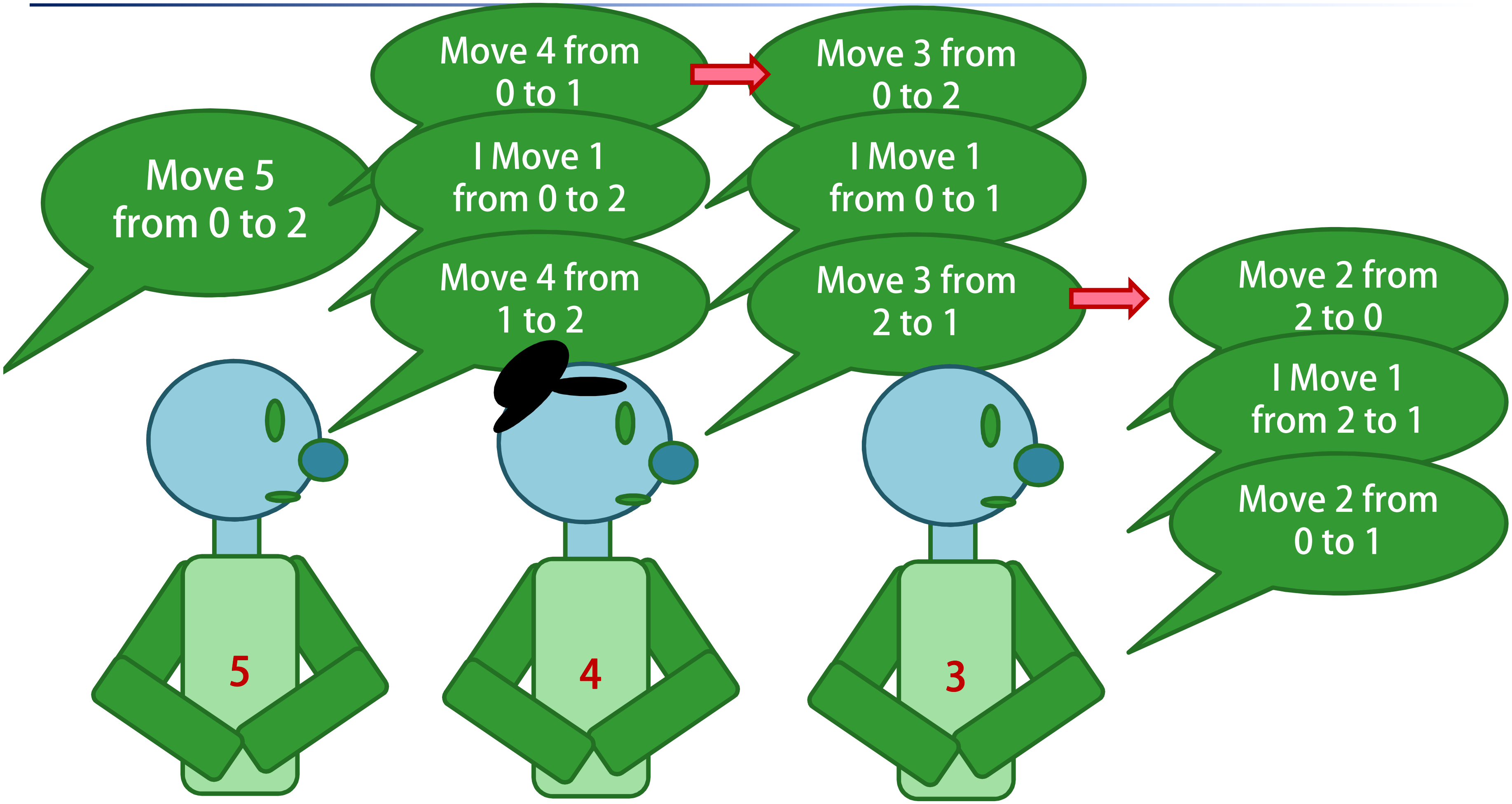


Extending from small to large

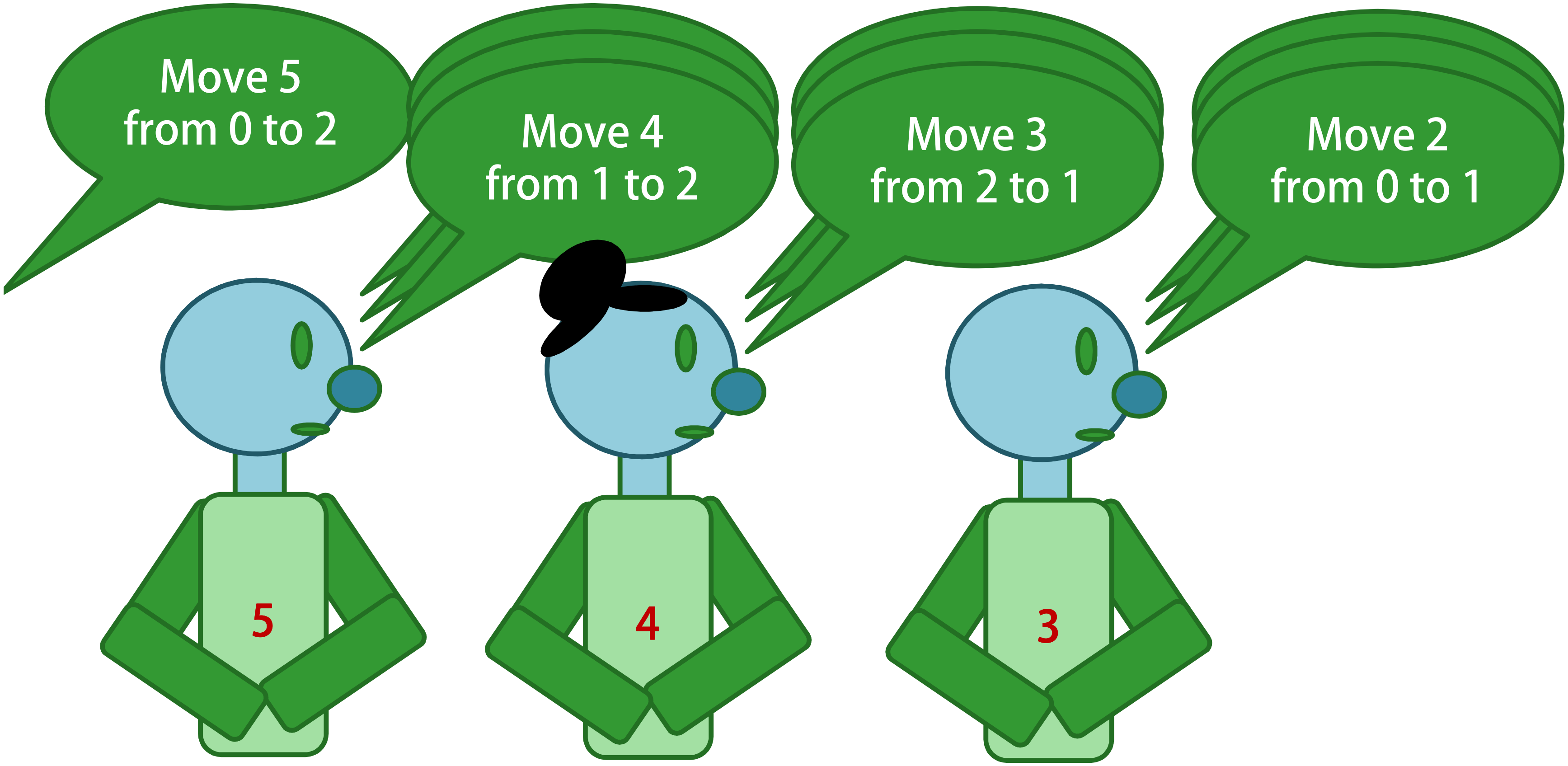
- Move n-1 to 2
- Move the nth disk to 3
- Move n-1 from 2 to 3



Human Recursion



Human Recursion



Expressing it in code

```
def movePiece(orig, dest):
    print "move from " + str(orig) + " to " + str(dest)
    x = raw_input()
```

```
def getOther(origin, dest):
    t = [0, 1, 2]
    t.remove(origin)
    t.remove(dest)
    return t[0]
```

Expressing it in code

```
def moveTower(origin, dest, n):
    if(n==1):
        movePiece(origin, dest)
    else:
        other = getOther(origin, dest)
        moveTower(origin, other, n-1)
        movePiece(origin, dest)
        moveTower(other, dest, n-1)
```

Visualizing the result

```
t0 = [5, 4, 3, 2, 1]
t1 = []
t2 = []
T = [t0,t1,t1]


def movePiece(orig, dest):
   tomove = T[orig][len(T[orig])-1]
   T[dest].append(tomove)
   T[orig].remove(tomove)
   print "move from " + str(orig) + " to " + str(dest)
   x = raw_input()
   print T[0]
   print T[1]
   print T[2]
```

Better

```
t0 = []
t1 = []
t2 = []
T = [t0,t1,t1]
#Don’t hard code the number of pieces. Initialize with a loop instead
def initialize(n):
    for x in xrange(1, n+1):
        t0.append(x)

def movePiece(orig, dest):
    tomove = T[orig][len(T[orig])-1]
    T[dest].append(tomove)
    T[orig].remove(tomove)
    print "move from " + str(orig) + " to " + str(dest)
    x = raw_input()
    print T[0]
    print T[1]
    print T[2]
```

The full code

Find it here: http://bit.ly/YzY6pj

Run this simpler version http://bit.ly/Yy0dpy on python tutor!

- http://www.pythontutor.com/
- It will really help you understand recursion

Human computers

WPA funded Mathematical Tables Project in New York City
David Alan Grier, "When Computers were Human"