Class Norms

- We are transforming our brains together
- Requires some trust and vulnerability
- “See something say something”
- “What is said here stays here, what is learned here leaves here”

Recursion and Induction

“The control of a large force is the same principle as the control of a few men: it is merely a question of deviding up their numbers” - Sun Zi, Art of War, 400CE

“To understand recursion, one must first understand recursion.” - Abraham Lincoln

- Common strategy for algorithms is “divide and conquer,” intentionally splitting a problem into smaller sub-problems.
- This is a certain type of recursion, described loosely as:
  - If the given instance of the problem can be solved directly, do it.
  - Otherwise, reduce it to one or more simpler instances of the **same** problem.

For example, instead of thinking of cleaning your entire apartment, often we start by saying “I will just clean my bedroom floor / bathroom counter / kitchen sink.” By completing enough smaller problems, we can solve the large problem.

Examples

Peasant Multiplication

x*y =

1. 0 if x=0
2. floor(x/2)*(y+y) if x is even
3. floor(x/2)*(y+y) + y if x is odd

Can also be expressed algorithmically as:

```
PeasantMultiply(x, y):

if x=0:
  return 0
else
  xNext = x // 2
  yNext = y+y
  prod = PeasantMultiply(xNext, yNext)
  if x is odd:
    prod = prod + y
  return prod
```

We can imagine the recursive call being done by us, or by someone else, even a magical being.

We will call this hypothetical magical being *the recursion fairy*.

Once we have defined the recursive call and the base case, we stop and let the recursion fairy handle the rest!

Example: Tower of Hanoi

Definition: ask audience to describe problem

Fun fact: puzzle was developed by a French mathematician at the same time as the original French invasion and colonization of Hanoi, and the establishment of French Indochina.

How to approach the problem?

- Can't solve until largest (nth) disk is moved
- So first, we must move all (n-1) smaller disks to the spare peg.
- Then, move largest disk to the destination.
- Now, all we have to figure out is. . . .

No! We are done!

Test with n=0, n=1, n=2, n=3

Can write recursive algorithm:

```
Hanoi(n, src, dst, tmp):

if n > 0:
  Hanoi(n-1), src, tmp, dst)
  move disk n from src to dst
  Hanoi(n-1, tmp, dst, src)
```

How to analyze?

Let T(n) denote the number of moves required to transfer n disks.

We have:

- T(0) = 0
- T(n) = 2T(n-1)+1

Compute first several values of T(n): 0, 1, 5, 11, 23, 47, 95. . .

Leads us to guess that $T(n) = 2^n - 1$, which happens to be correct.

If time: Recursion trees and merge sort