Recursion!!!

First let's review proofs by induction:

Idea: want to prove a statement T that depends on an integer n. So $T(n) \equiv$ truth for n.

Idea:

1. Show T explicitly for some small value of n, call it n_0.
2. Show that in general, if $T(n-1)$ is true, then $T(n)$ must also be true.
3. Conclude $T(n)$ true $\forall\ n \geq n_0$.

Example: Prove $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$.

1. $n_0 = 1$. $\sum_{i=1}^{1} i = 1 = \frac{1(1+1)}{2}$ ✓

2. if $\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$, then

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}.$$

$$\sum_{i=1}^{n} i = \left(\sum_{i=1}^{n-1} i\right) + n$$

$$= \frac{n(n-1)}{2} + n$$

$$= \frac{n(n-1) + 2n}{2}$$

$$= \frac{n^2 + n}{2} = \frac{n(n+1)}{2} \quad \checkmark$$

(contrast with other proofs you've seen, say, in geometry. Format: If A, then B.

$A \Rightarrow S_1$ (SAS theorem)

$S_1 \Rightarrow S_2$ (algebra.)

$S_2 \Rightarrow S_3$ (")

$S_3 \Rightarrow B$ (") $\checkmark$

Notice that there were a fixed # of implications ($\Rightarrow$). Induction gives you a way to write proofs with an arbitrary # of implications:

say $n_0 = 1$

$T(1) \Rightarrow T(2) \Rightarrow T(3) \Rightarrow \cdots \Rightarrow T(n)$

① ② ② ② ②

③

What does this have to do w/ programming??

The connection is as follows:

We want to write a program that computes

Some function f.

T(n) ≡ "our program sets the right answer on any input of size n"

New programming technique: recursion:

① Hard-code answer for small inputs:

```
int h(n, ...) {
    if (n==1)
      //return right answer...
```

② Assuming your program works for smaller inputs, build the solution to your input of size n. This may involve calling your own function! (on smaller inputs)

③ Celebrate?

Warm up example: Compute n!

```
int fac(int n) {
  // ①
  if(n==0) return 1;
  // ②
  return fac(n-1) * n;
}
```

Here's a trace