Recursion

Similar to mathematical induction.

What is induction? It's a proof technique that allows for arbitrary #'s of implications ("steps").

Thm: $A \Rightarrow E$

Proof:

$A \Rightarrow B$ (axiom 1)
$B \Rightarrow C$ (algebra...)
$C \Rightarrow D$ (triangle-ineq)
$D \Rightarrow E$ (more algebra).

Proof had a fixed # of "$\Rightarrow$"

Induction: kind of meta. It is machinery that can construct proofs with arbitrarily long sequences of "$\Rightarrow$".

How does it work: usually used to prove statements parameterized by natural #'s $(1, 2, 3 \ldots)$.
Denote the truth of the n^{th} statement by $t(n)$.
Here are the steps for a proof by induction:

1. Explicitly demonstrate $t(1)$ (truth of first statement).
2. Show that in general, $t(n-1) \Rightarrow t(n)$.

③ Conclude truth for all n:

$t(1) \overset{①}{\Rightarrow} t(2) \overset{②}{\Rightarrow} t(3) \overset{②}{\Rightarrow} \cdots \overset{②}{\Rightarrow} t(n) \overset{③}{}$

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

Proof:

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

② Assuming $\sum_{i=1}^{n-1} i = \frac{(n-1)n}{2}$, need to show $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$.

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

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

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

$$= \frac{n(n+1)}{2}$$ ✓

What does this have to do with programming?

Usually $t(n) \equiv$ "my program computes the right value on inputs of size n".

Then we write a self-referencing program as follows:

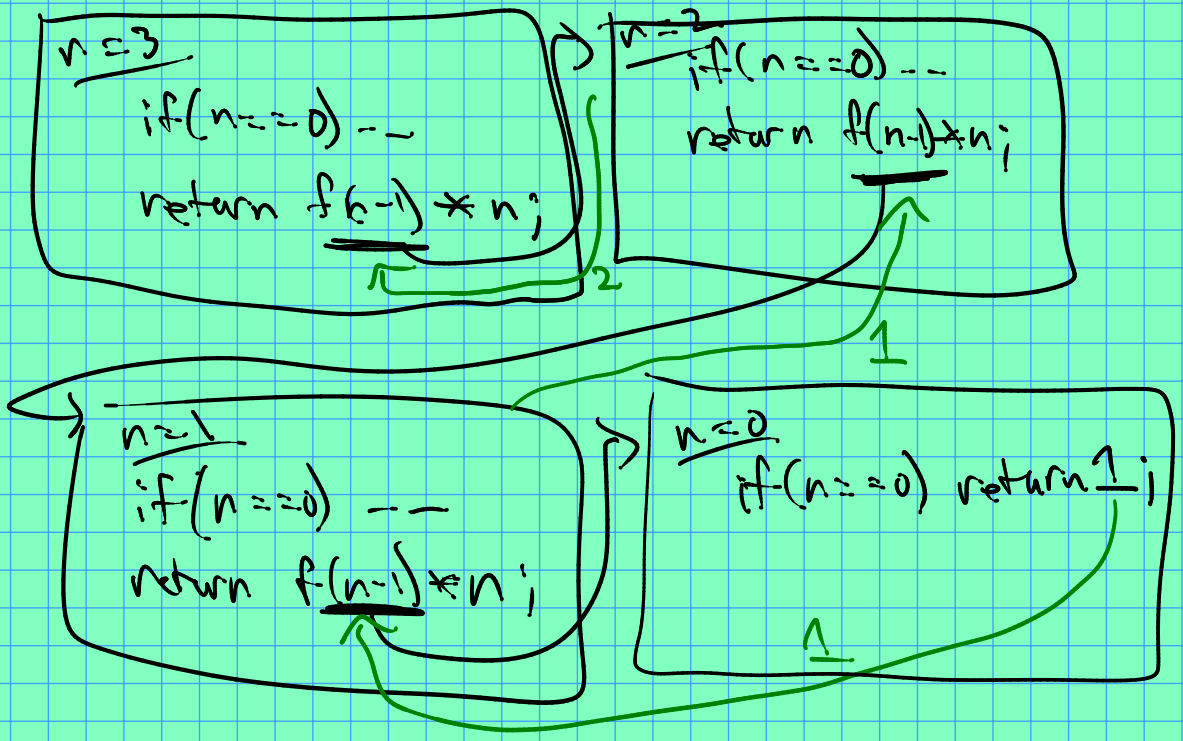
```
int f(int n) {
    if (n == 1) return right-answer;
```

```
// otherwise, build f(n) from
// f(n-1), f(n-2) ...
```

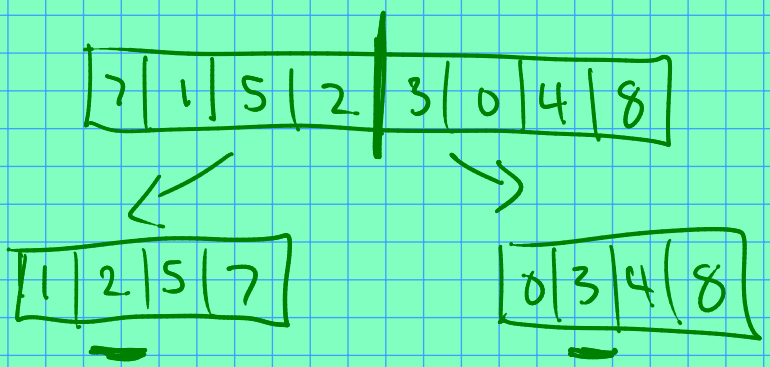
Example: compute n!

```
int f(int n) {
    if(n == 0) return 1;
    return f(n-1)*n;
}
```

Let's trace it on input 3:



Recursive sorting:



0 1

```
vector<int> sort(vector<int>& v) {
    if(v.size() < 2) return v;
    //else
    // break v into left & right
    vector<int> vleft(0, v.size()/2);
    vector<int> vright(v.size()/2+1, v.size()-1);
    // sort vleft & vright by "recursive magic"
    vleft = sort(vleft);
    vright = sort(vright);
    // merge results!
    vector<int> sorted;
    int i=0, j=0;
    while(i< vleft.size() && j<vright.size())
        ;
        ;

    return sorted;
}
```