New Topic: Writing <u>Recursive</u> Functions (l5.pdf)

Very similar to (basically a special case of) proofs by induction:

Usually want to prove a parameterized statement, $T(n)$. E.g. $\sum_{i=0}^{n} r^i = \frac{1-r^{n+1}}{1-r}$ (say $r>0$)

Want to show $T(n)$ <u>true for all $n \geq 0$</u>.

Induction:

① prove statement explicitly for small value of n.

$$T(0): \sum_{i=0}^{0} r^i = r^0 = 1 = \frac{1-r^{0+1}}{1-r} \checkmark$$

② Assume $T(n')$ is true for all $n' < n$. Use that to show $T(n)$ must also be true.

$$\sum_{i=0}^{n} r^i = r^n + \sum_{i=0}^{n-1} r^i$$

$$= r^n + \frac{1-r^n}{1-r}$$

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

$$= \frac{\cancel{r^n} - r^{n+1} + 1 - \cancel{r^n}}{1-r}$$

$$= \frac{1 - r^{n+1}}{1-r} \checkmark$$

Summary:

1. Prove "base case" (maybe $T(0)$) explicitly.
2. Prove that $T(n-1) \Rightarrow T(n)$
3. Conclude $T(n)$ true for all $n \geq 0$.

$$\underset{①}{T(0)} \underset{②}{\Rightarrow} T(1) \underset{②}{\Rightarrow} T(2) \underset{②}{\Rightarrow} T(3) \Rightarrow \cdots$$

Contrast w/ "normal" proofs from maybe your geometry class: want $A \Rightarrow E$.

Proof: $A \Rightarrow B$ (SAS theorem)

$B \Rightarrow C$ (arithmetic)

$C \Rightarrow D$ (angle formula)

$D \Rightarrow E$ (...)

Note: in a sense, inductive proofs are kind of like proof generators... from pieces 1, 2 a traditional proof could be derived for any value of n, but there is no bound on the length!

Recursion

Will usually have the following shape:

```
f(n)
{   if(n<1) {        // 1 for induction
        return <<right answer>>;
    }
    //else ...
    // we pretend our function f
    // (which we're not done writing yet!!)
    // works as it should on any smaller input.
    // call our own f... assemble results to
    // get our answer:
    a = f(n-1);  b = f(n-3);

    return <<some expression of a,b ...>>;
            // == f(n)
}
```

⊗ Recursive functions can almost be thought of as <u>inductive proofs of their own correctness.</u>

$$T(n) \equiv \text{"my function works on any input of size } n\text{."}$$

Lame example: $f(n) = n! = 1 \cdot 2 \cdot 3 \cdots n$.

```
size_t f(size_t n)
{
    if(n==0) return 1;
    // now assume f works for any
    // input < n. E.g. n-1.
    size_t a = f(n-1); // should be (n-1)!
    return n*a; // n·(n-1)! = n!
}
```