Reducing a problem $X$ to a problem $Y$:

describe/design an algorithm for $X$ that uses an algorithm for $Y$ as a "black box" or "subroutine"
does not depend on how alg for $Y$ works
uses only what is known fast for $Y$'s alg

similar to proving theorems using lemmas
special reductions:
induction + recursion
Let $n$ be a positive integer.

A divisor $d$ of $n$ is an integer $d$ s.t. $n/d$ is an integer.

$n$ is prime if it has exactly two divisors: $n$ and $1$.

$n$ is composite if more than two divisors.

Thm: Every integer $n > 1$ has a prime divisor.
Suppose exists an integer > 1 with no prime divisor.

Let \( n \) be the smallest integer > 1 with no prime divisor. \( n \) divides itself, but \( n \) has no prime divisors \( \implies \)

\( n \) is not prime.

\( \exists \) a divisor \( d \) of \( n \) s.t. \( 1 < d < n \).

\( n \) is smallest counter example

So a prime \( p \) divides \( d \).
\[ d/p \text{ is an integer} \]

\[ \Rightarrow \left( \frac{n}{d} \right) \cdot \left( \frac{d}{p} \right) = \frac{n}{p} \text{ is an integer} \]

so \( p \) divides \( n \) \( \downarrow \)

\text{contradiction!}
Direct proof:
Let \( n \) be an arbitrary integer \( \geq 1 \).
Assume for every integer \( k \)
s.t. \( 1 \leq k < n \), integer \( k \) has
a prime divisor.
Suppose \( n \) is prime, it is
its own prime divisor.
Otherwise, \( \exists \) divisor \( d \) s.t. \( 1 < d = n \)
By assumption, \( d \) has a
prime divisor \( p \).
\( d/p \) is an int \( \Rightarrow \) \( \exists \lambda \in \mathbb{Z} \).
\( (d/p) = n/p \)
is an integer.
So \( p \) is a prime divisor of \( n \).
proof by induction

induction hypothesis; assuming no strictly smaller counterexamples

Case cases: argue directly without using hypothesis (n is prime)

Inductive cases: the other cases
**Theorem:** $P(n)$ for every positive integer $n$.

**Proof by induction:** Let $n$ be an arbitrary positive integer. Assume that $P(k)$ is true for every positive integer $k < n$. There are several cases to consider:

- Suppose $n$ is $\ldots$ blah blah blah $\ldots$
  
  Then $P(n)$ is true.

- Suppose $n$ is $\ldots$ blah blah blah $\ldots$
  
  The inductive hypothesis implies that $\ldots$ blah blah blah $\ldots$
  
  Thus, $P(n)$ is true.

In each case, we conclude that $P(n)$ is true.

---

**Recipe:**

1) Write down template

2) Think big

3) Look for holes (base cases)

4) Rewrite everything

---

\[
\text{trust claim is true for } k < n
\]
DO NOT:

1) Do not make the hypothesis for $k=n-1$ only.

2) Do not assume for $n$ and prove for $n+1$. Don’t build up, instead reach down.
Recursion: Given a problem instance.

1) try to reduce it to one or more simpler instances of the same problem (smaller problem size).

2) if you can’t reduce, solve the instance directly (base cases).

Trust the Recursion Fairy can solve the simpler instances.
peasant multiplication:

\[ x \cdot y = \begin{cases} 
  0 & \text{if } x = 0 \\
  \left\lfloor \frac{x}{2} \right\rfloor \cdot (y+y) & \text{if } x \text{ is even} \\
  \left\lfloor \frac{x}{2} \right\rfloor \cdot (y+y)+y & \text{if } x \text{ is odd} 
\end{cases} \]

just need to know addition and halving

```
PeasantMultiply(x, y):
    if x = 0
        return 0
    else
        x' ← \lfloor x/2 \rfloor
        y' ← y + y
        prod ← PeasantMultiply(x', y')  ⟨ ⟨Recurse!⟩⟩
        if x is odd
            prod ← prod + y
        return prod
```
Correctness of recursive algorithms follows from induction.

Given $x+y$.

If $x=0$, $x\cdot y=0$.

O.w. $x'=\lceil x/2 \rceil < x$.

Assume $PM(k,y)$ is correct for $0 \leq k < x$.

So $PM(x',y')$, it is correct.

Rest of alg follows formula.