Reductions

Reducing a problem \( X \) to another problem \( Y \) means having an algorithm for \( X \) use an algorithm for \( Y \) as a "black-box" or subroutine.
Uses what + how fast of Y only.

Like using a lemma in math.
Let $n$ be a positive integer. A divisor of $n$ is a positive integer $p$ s.t. $n/p$ is an integer.

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

$n$ is composite if it has $>2$ divisors.

$1$ is neither.
Thm: Every integer \( n \) greater than 1 has a prime divisor.

two proof techniques:

Direct proof: Let \( n > 1 \)

\[ \ldots \]

\( n \) has a prime divisor

Proof by contradiction:
Assume some int \( n > 1 \) has no prime divisor

\[ \ldots \]

We have a contradiction.
Proof by contradiction. Assume there is some integer \( n \) with no prime divisor. \( n \) divides itself, so \( n \) has no prime divisor, so \( n \) is not prime. Thus, exists at least one divisor \( d \) where \( 1 < d < n \). \( n \) has no prime divisors, so \( d \) is not prime. Thus \( d \) has a divisor \( d' \) where \( 1 < d' < d \).
Because \( d / d' \) is an integer,

\[ n / d' = (n / d) \cdot (d / d') \]

is a divisor of \( n \).

So \( d' \) is a divisor of \( n \).

So \( d' \) is not prime.

So \( d' \) has a divisor \( d'' \) where \( 1 < d'' < d' \).

So \( d'' \) is a divisor of \( n \)...

STOP
Another try...

Proof by \underline{smallest} counter example.

Assume some integer \( n > 1 \) has no prime divisor \( 1 \) and let \( n \) be the smallest example. \( n \) divides itself, \( \therefore n \) has no prime divisor, so \( n \) is not prime.

Thus, exists at least one divisor \( d \) where \( 1 < d < n \).

\( n \) was the smallest counterex, so \( d \) has a prime divisor \( p \).
$n/p = (n/d) . (d/p)$ is an integer.

So $p$ is a prime divisor of $n$.

So there are no counterexamples.
Direct proof: Let \( n \) be an integer \( \geq 1 \).
Assume for all integers \( k \) s.t. \( 1 \leq k < n \), \( k \) has a prime divisor.
If \( n \) is prime, it is its own prime divisor.

\( \forall \) otherwise \( \forall \) otherwise

So it has a divisor \( d \) s.t. \( 1 < d < n \).

By assumption \( d \) has a
prime divisor \( p \).

\( (n/p) = (n/d), (d/p) \) is an integer, so \( p \) is a prime divisor of \( n \).

Was a proof by induction.

Induction hypothesis (IH) assume theorem true for strictly smaller integers.
Inductive case: Using the IH.

Base case: Not using the IH. May be an infinite # of them!
Recipe:

1) Write down the template.

![Theorem]

**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 ... *blah blah blah* ...
  
  Then $P(n)$ is true.

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

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

2) Think big. Start with Inductive step.

3) Fill in the gaps (base cases).

4) Rewrite!
DO NOT:
1) Assume only on n-1.
Do assume for all \( k < n \).
2) Do a proof for "n+1".
Recursion:

Write an algorithm to solve problem \( X \) that...

1) Reduce large inputs to smaller inputs of \( X \).

2) Solve other instances directly (base cases).

Treat the recursive calls as black-box reductions.

The Recursion Fairy solves them.
- move one disk at a time
- never place a larger disk on a smaller one
- get all disks from left to right
Observations: 1) biggest disk cannot prevent others from moving

Algorithm:
1) get smaller n-1 disks of biggest one, somehow...
2) move biggest disk to destination
3) put n-1 smaller disks on it, somehow...
Hanoi (n, src, dst, tmp):
move n disks from src to dst, using tmp as temp space (disks go small to big 1 to n)

\[
\text{Hanoi}(n, \text{src}, \text{dst}, \text{tmp}):
\begin{align*}
\text{if } n > 0 & \quad \text{Hanoi}(n-1, \text{src}, \text{tmp}, \text{dst}) \quad \text{\textit{\small (Recurse!)} } \\
\text{move disk } n \text{ from src to dst } & \quad \text{Hanoi}(n-1, \text{tmp}, \text{dst}, \text{src}) \quad \text{\textit{\small (Recurse!)} }
\end{align*}
\]
\( T(n) \): \# moves for \( n \) disks

\[ T(0) = 0 \]

\[ T(n) = 2T(n-1) + 1 \quad (n > 0) \]

**Thm:** \( T(n) = 2^n - 1 \)

**Proof:** Assume \( T(k) = 2^k - 1 \) for all \( k \leq n \).

**Is** \( n = 0 \), \( T(n) = T(0) = 0 = 2^0 - 1 \)

**Is** \( n > 0 \),

\[ T(n) = 2T(n-1) + 1 \]

\[ = 2 \cdot (2^{n-1} - 1) + 1 \]

\[ = 2^n - 1 \] \( \checkmark \)