Reductions

Reducing a <u>problem</u> X to another <u>problem</u> Y means having an algorithm for X use an algorithm for Y as a "black-box" or subroutine.

Uses <u>what</u> & <u>how fast</u> of Y only.

Like using a lemma in math

Let n be a positive integer

A <u>divisor</u> of n is a pos. integer p s.t. n/p is an integer.

n is <u>prime</u> if it has exactly two divisors, 1 & n.

n is <u>composite</u> 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 $n > 1$

...

n has a prime divisor

Proof by contradiction:

Assume some int $n > 1$ has no prime divisor

...

We have a contradiction.

Proof by contradiction.

Assume there is some int n with no prime divisor.

n divides itself, & 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'

$n/d' = (n/d)\cdot(d/d')$

is an integer.

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 <u>smallest</u> counter example.

Assume some integer > 1 has no prime divisor & let n be the smallest example.

n divides itself, & 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.

$n/p = (n/d) \cdot (d/p)$ is an integer.

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

contradiction

So there are no counter examples.

Direct proof: Let n be an integer > 1.

Assume for all integers k s.t. $1 < k < n$, k has a prime divisor.

If n is prime, it is its own prime divisor.

otherwise

o.w. n is composite

So it has a divisor d s.t. $1 < d < n$.

By assumption d has a

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

Was a proof by induction.

Induction hypothesis (IH) assump theorem true for strictly smaller integers.

Inductive case: Using the I.H.

Base case: Not using the I.H. May be an infinite # of them!

Recipe:

1) Write down the template.

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:

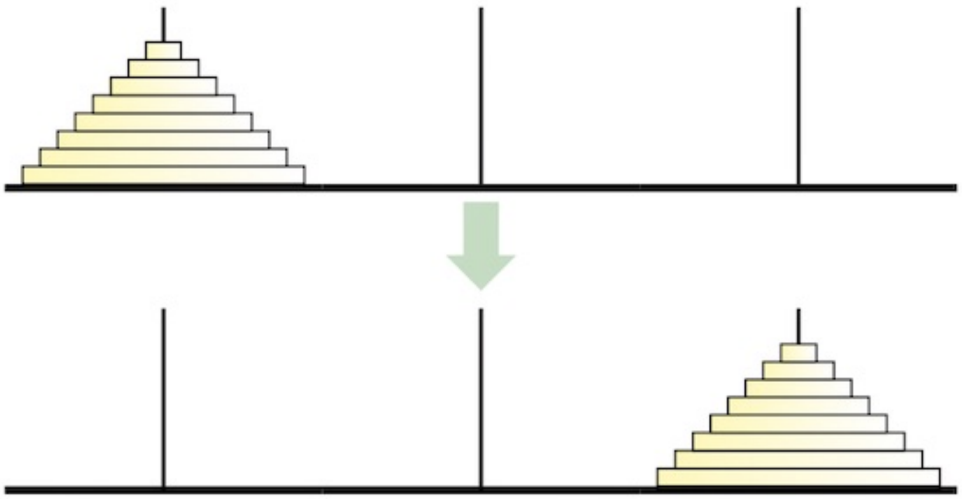
Write an algorithm to solve problem X that...

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

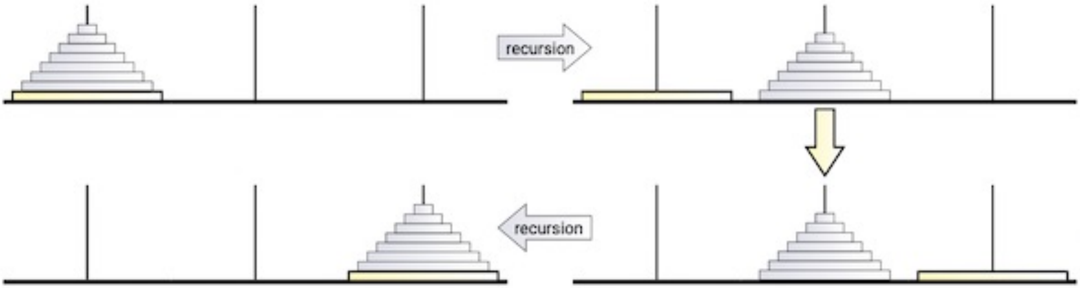
Observations: 1) biggest disk cannot prevent others from moving

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)

```
HANOI(n, src, dst, tmp):
  if n > 0
     HANOI(n − 1, src, tmp, dst)     ⟨⟨Recurse!⟩⟩  T(n-1)
     move disk n from src to dst     1
     HANOI(n − 1, tmp, dst, src)     ⟨⟨Recurse!⟩⟩
```

T(n-1)

$T(n)$: # moves for n disks

$T(0) = 0$

$T(n) = 2T(n-1) + 1$

$(n > 0)$

Thm: $T(n) = 2^n - 1$

Proof: Assume $T(k) = 2^k - 1$

for all $k < n$.

If $n = 0$, $T(n) = T(0) = 0 = 2^0 - 1$

$= 2^n - 1$ ✓

If $n > 0$,

$$T(n) = 2T(n-1) + 1$$

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

$$= 2^n - 1 \quad ✓$$