RECURSIVE ALGORITHMS

DISCRETE STRUCTURES II

DARRYL HILL

BASED ON THE TEXTBOOK:

DISCRETE STRUCTURES FOR COMPUTER SCIENCE: COUNTING, RECURSION, AND PROBABILITY

BY MICHIEL SMID

A Message Problem

As computer scientists, we should be able to write and analyze recursive algorithms.

Let $n \geq 4$ be an integer and consider n people:

$$P_1, P_2, \ldots, P_n$$

Every person P_i has a message M_i that they want to send to everyone else.

To begin with, each person knows their own message as in the table:

$\boldsymbol{P_1}$	$\boldsymbol{P_2}$	…	$\boldsymbol{P_{n-1}}$	$\boldsymbol{P_n}$
$\boldsymbol{M_1}$	$\boldsymbol{M_2}$	…	$\boldsymbol{M_{n-1}}$	$\boldsymbol{M_n}$

After running our algorithm we want:

$\boldsymbol{P_1}$	$\boldsymbol{P_2}$	…	$\boldsymbol{P_{n-1}}$	$\boldsymbol{P_n}$
$\boldsymbol{M_1, \ldots, M_n}$	$\boldsymbol{M_1, \ldots, M_n}$	…	$\boldsymbol{M_1, \ldots, M_n}$	$\boldsymbol{M_1, \ldots, M_n}$

Assume that P_i and P_j can connect over a VPN and share all messages.

If P_i connects to P_j, then P_i learns all of P_j's messages, and P_j learns all of P_i's messages.

A Message Problem

Let $n \geq 4$ be an integer and consider

$$P_1, P_2, \ldots, P_n$$

Every person P_i has a message M_i that they want to send to everyone else.

How do we get from Table 1

P_1	P_2	…	P_{n-1}	P_n
M_1	M_2	…	M_{n-1}	M_n

to Table 2

P_1	P_2	…	P_{n-1}	P_n
$M_1, \ldots, M_n$	$M_1, \ldots, M_n$	…	$M_1, \ldots, M_n$	$M_1, \ldots, M_n$

using the fewest connections?

If we use a "brute force" algorithm we can have everyone message everyone else. Then there are:

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

connections. Let's try a small example to see if we can do better.

P_1	P_2	P_3	P_4
M_1	M_2	M_3	M_4

A Message Problem

Let $n = 4$.

Consider the following sequence of connections.

P_1	P_2	P_3	P_4
M_1	M_2	M_3	M_4

1. P_1 connects to P_2:

P_1	P_2	P_3	P_4
M_1, M_2	M_1, M_2	M_3	M_4

2. P_3 connects to P_4:

P_1	P_2	P_3	P_4
M_1, M_2	M_1, M_2	M_3, M_4	M_3, M_4

3. P_1 connects to P_3:

P_1	P_2	P_3	P_4
$M_1, M_2,$ M_3, M_4	M_1, M_2	$M_1, M_2,$ M_3, M_4	M_3, M_4

4. P_2 connects to P_4:

P_1	P_2	P_3	P_4
$M_1, M_2,$ M_3, M_4	$M_1, M_2,$ M_3, M_4	$M_1, M_2,$ M_3, M_4	$M_1, M_2,$ M_3, M_4

A Message Problem

For $n = 4$ it took 4 connections to spread all messages to everyone.

P_1	P_2	P_3	P_4
M_1, M_2, M_3, M_4	M_1, M_2, M_3, M_4	M_1, M_2, M_3, M_4	M_1, M_2, M_3, M_4

Using brute force would be $\binom{4}{2} = 6$ messages.

Now that we can solve the "base case" of our algorithm we can attempt to solve this problem recursively.

We can solve for n by ***assuming*** we can solve recursively for $n - 1$, and using this solution to solve the main problem.

(Solving the base case makes this a reasonable assumption.)

1. At the start, each person P_i knows only the message M_i.

P_1	P_2	…	P_{n-1}	P_n
M_1	M_2	…	M_{n-1}	M_n

A Message Problem

For $n = 4$ it takes 4 connections to spread all messages to everyone.

1. At the start, each person P_i knows only the message M_i.

P_1	P_2	…	P_{n-1}	P_n
M_1	M_2	…	M_{n-1}	M_n

2. P_{n-1} connects to P_n.

P_1	P_2	…	P_{n-1}	P_n
M_1	M_2	…	M_{n-1}, M_n	M_{n-1}, M_n

3. Recursively solve for people P_1 through P_{n-1} (we've assumed we can do this). Only P_n does not know every message.

P_1	P_2	…	P_{n-1}	P_n
$M_1, \dots, M_n$	$M_1, \dots, M_n$	…	$M_1, \dots, M_n$	M_{n-1}, M_n

4. Have P_{n-1} connect to P_n again. Now P_n knows everything.

P_1	P_2	…	P_{n-1}	P_n
$M_1, \dots, M_n$	$M_1, \dots, M_n$	…	$M_1, \dots, M_n$	$M_1, \dots, M_n$

So if the recursive call works, we can solve the problem.
Also, the base case does work (we solved it).

A Message Problem

Algorithm MESSAGE(n):

if $n = 4$:
  P_1 messages P_2;
  P_3 messages P_4;
  P_1 messages P_3;
  P_2 messages P_4;
else:
  P_{n-1} messages P_n;
  MESSAGE(n-1);
  P_{n-1} messages P_n;

We have shown it is correct using informal induction.

But what is the runtime? We can write it as a recursive function.

$C(4) = 4$: this is the base case.

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

We can expand it out, guess at a closed form, then prove by induction.

$$C(5) = 2 + 4 = 6$$
$$C(6) = 2 + 6 = 8$$
$$C(7) = 2 + 8 = 10$$

We guess $C(n) = 2n - 4$.

A Message Problem

Algorithm MESSAGE(n):

if $n = 4$:
  P_1 messages P_2;
  P_3 messages P_4;
  P_1 messages P_3;
  P_2 messages P_4;
else:
  P_{n-1} messages P_n;
  MESSAGE(n-1);
  P_{n-1} messages P_n;

$C(4) = 4$
$C(n) = 2 + C(n-1)$

Our guess: $C(n) = 2n - 4$

Base case: $C(4) = 2(4) - 4 = 4$

So the base case holds.

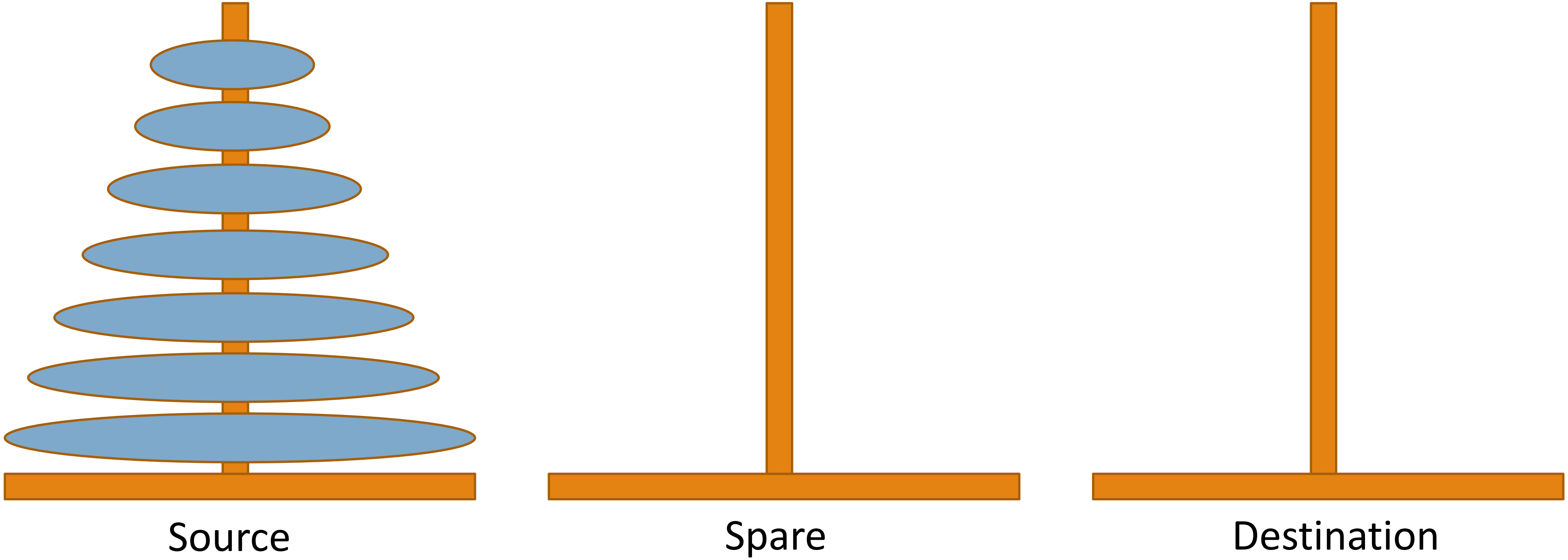
Inductive hypothesis:

$$\begin{aligned} C(n-1) &= 2(n-1) - 4 \\ &= 2n - 6 \end{aligned}$$

$$\begin{aligned} C(n) &= 2 + C(n-1) \\ &= 2 + 2n - 6 \\ &= 2n - 4 \end{aligned}$$

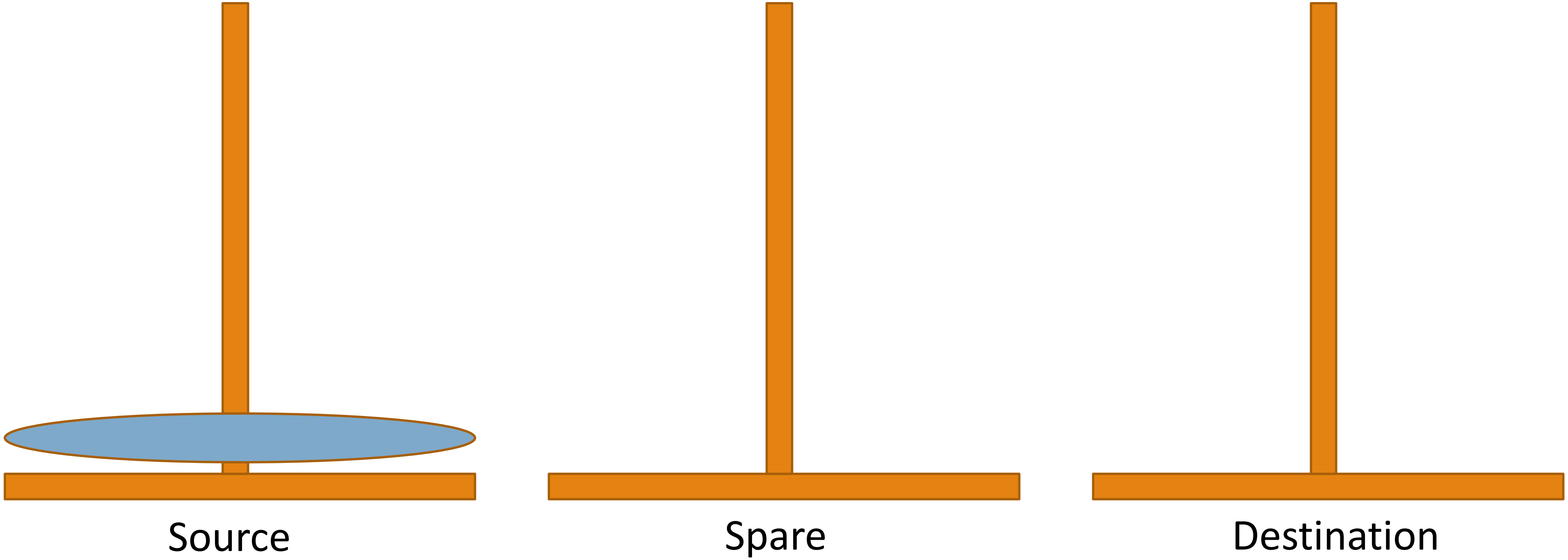
Towers of Hanoi

Move all disks to third peg.
Never put a larger disk on a smaller disk.
Move disks one at a time.



Towers of Hanoi

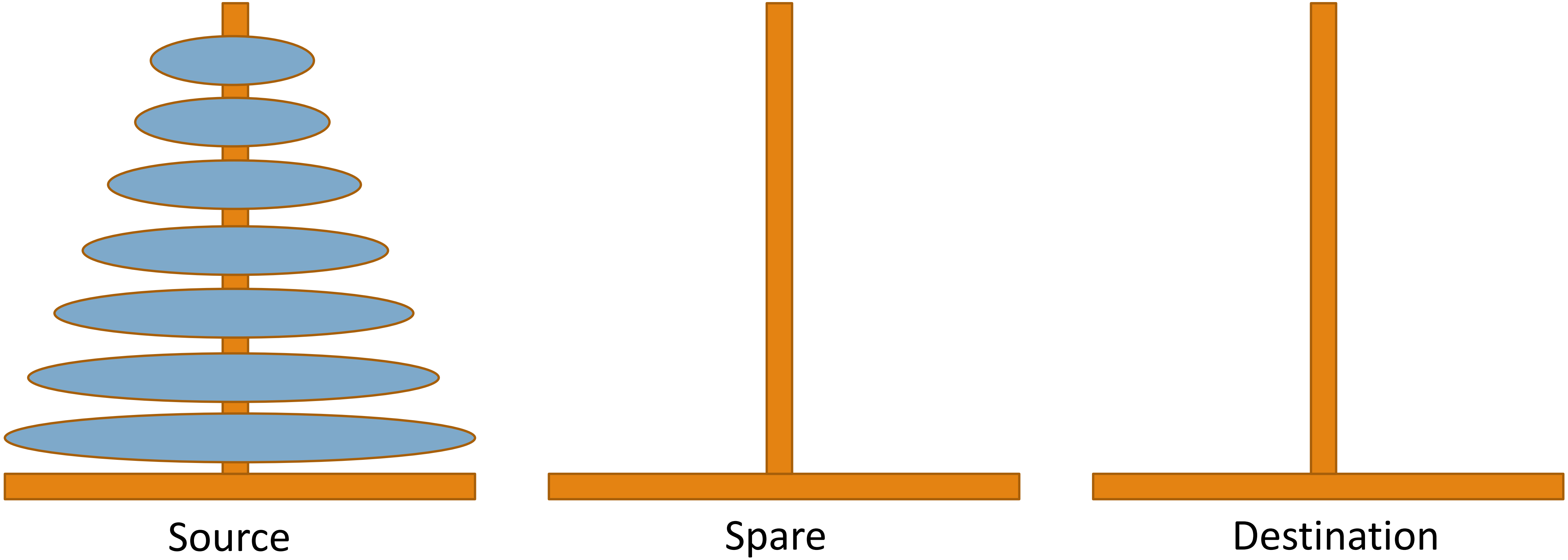
Move all disks to third peg.
Never put a larger disk on a smaller disk.
Move disks one at a time.



Solver recursively. Base case?
1 disk, move from peg 1 to peg 3.

Towers of Hanoi

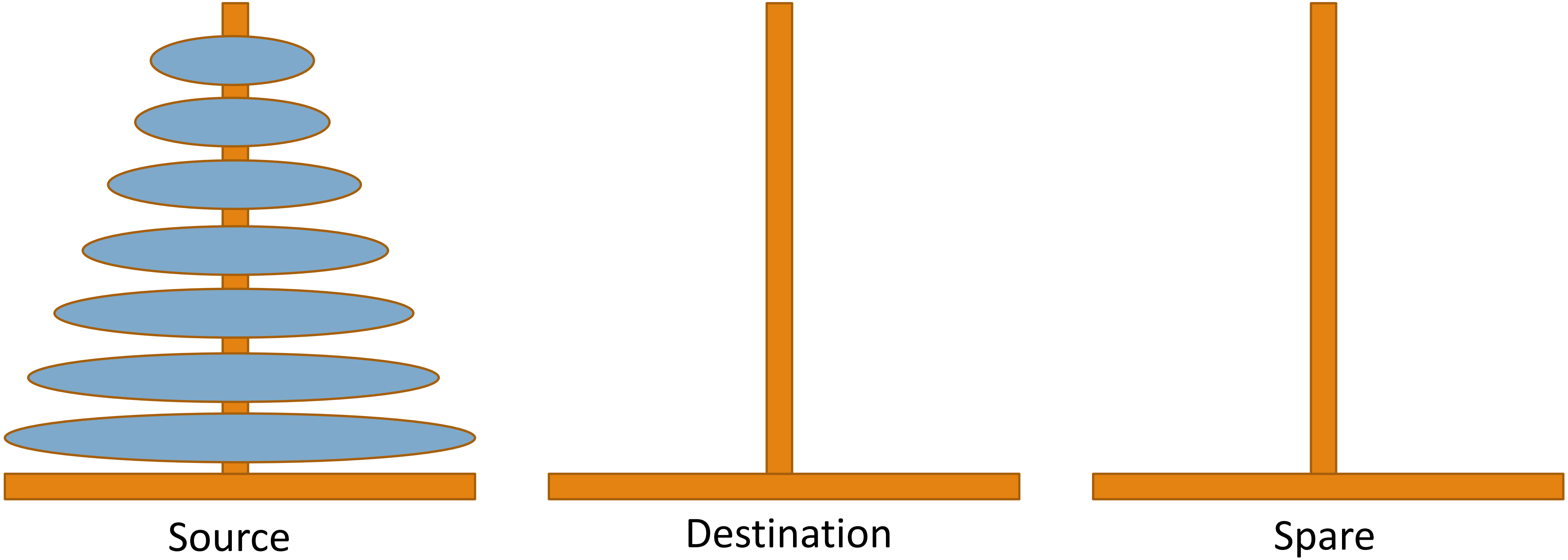
Move all disks to third peg.
Never put a larger disk on a smaller disk.
Move disks one at a time.



Assume we can solve for $n - 1$ disks.

Towers of Hanoi

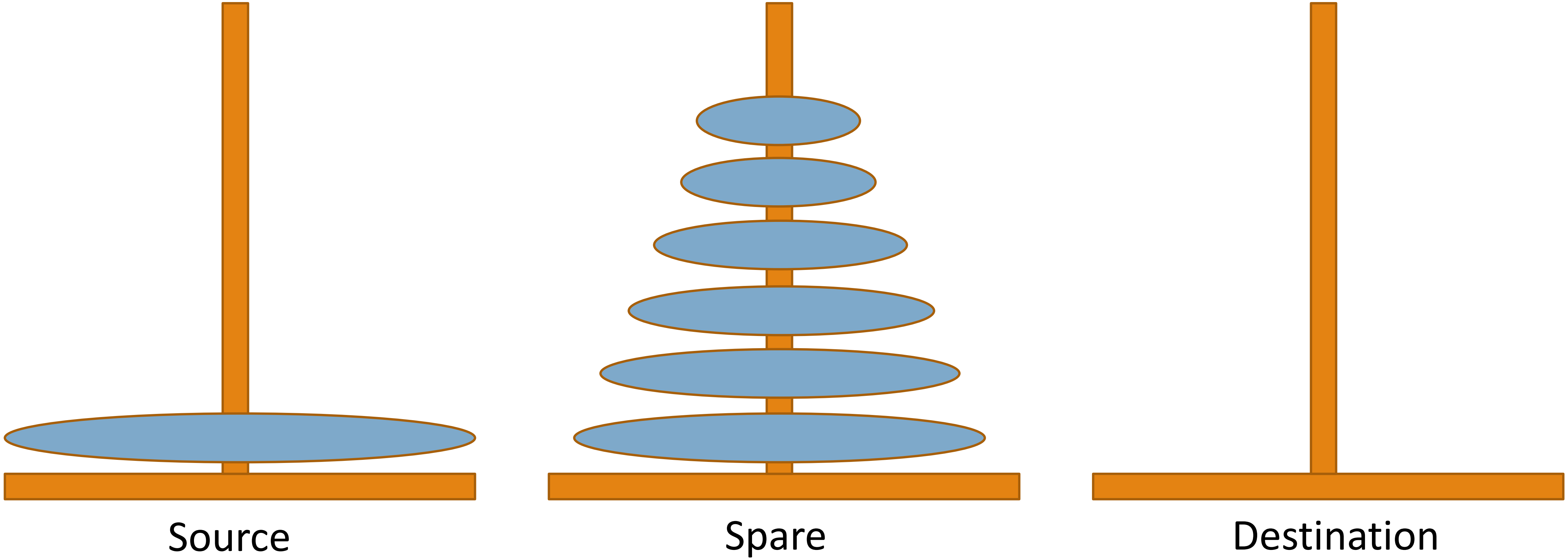
Move all disks to third peg.
Never put a larger disk on a smaller disk.
Move disks one at a time.



Assume we can solve for $n - 1$ disks.

Towers of Hanoi

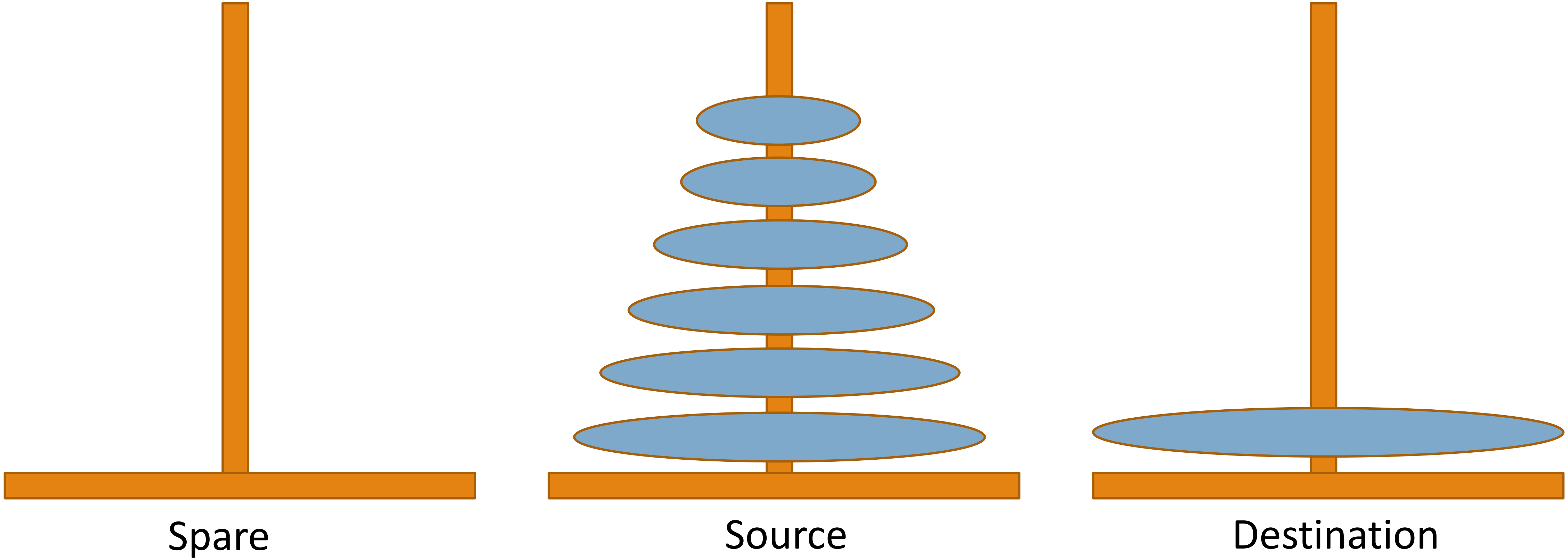
Move all disks to third peg.
Never put a larger disk on a smaller disk.
Move disks one at a time.



Assume we can solve for $n-1$ disks.

Towers of Hanoi

Move all disks to third peg.
Never put a larger disk on a smaller disk.
Move disks one at a time.



Assume we can solve for $n - 1$ disks.

Algorithm HANOI(source, dest, spare, n):

if $n = 1$:
  Move1Disk(source, dest**)**;
else:
  HANOI(source, spare, dest, n-1);
  Move1Disk(source, dest**)**;
  HANOI(spare, dest, source, n-1);

Let $H(n)$ be the number of moves to complete the puzzle on n disks.

Towers of Hanoi

$$H(n) = H(n-1) + 1 + H(n-1)$$
$$H(n) = 2 \cdot H(n-1) + 1$$
$$H(n) = 2 \cdot (2 \cdot H(n-2) + 1) + 1$$
$$H(n) = 4 \cdot H(n-2) + 2 + 1$$
$$H(n) = 8 \cdot H(n-3) + 4 + 2 + 1$$
…
$$H(n) = 2^n \cdot H(1) + \sum_{i=0}^{n-1} 2^i$$
$$H(n) = 2^n + ???$$

Towers of Hanoi

$$\begin{aligned}
s_n &= ar^0 + ar^1 + ar^2 + \cdots ar^{n-1} \\
rs_n &= \qquad\quad ar^1 + ar^2 + \cdots ar^{n-1} + ar^n \\
\\
s_n - rs_n &= ar^0 - ar^n \\
s_n(1-r) &= a(1-r^n) \\
s_n &= a\left(\frac{1-r^n}{1-r}\right)
\end{aligned}$$

Towers of Hanoi

$$H(n) = H(n-1) + 1 + H(n-1)$$
$$H(n) = 2 \cdot H(n-1) + 1$$
$$H(n) = 2 \cdot (2 \cdot H(n-2) + 1) + 1$$
$$H(n) = 4 \cdot H(n-2) + 2 + 1$$
$$H(n) = 8 \cdot H(n-3) + 4 + 2 + 1$$
$$\dots$$
$$H(n) = 2^n \cdot H(1) + \sum_{i=0}^{n-1} 2^i$$
$$H(n) = 2^n + \left(\frac{1-2^n}{1-2}\right)$$
$$H(n) = 2^n + 2^{n-1}$$
$$H(n) = 3 \cdot 2^{n-1}$$

$$s_n = a\left(\frac{1-r^n}{1-r}\right)$$
$$r = 2$$
$$a = 1$$

Euclid's Algorithm

Greatest common divisor.

$a \geq 1, b \geq 1, \gcd(a, b) =$ largest integer that divides both a and b.

Example: $\gcd(75,45) =$

Common divisors: $1, 3, 5, 15$

$\gcd(a, a) = a$

How do we find gcd of large numbers?

$a = 371\ 435\ 805$
$b = 137\ 916\ 675$

$\gcd(a, b) =$

Euclid's Algorithm

Greatest common divisor.

$a \geq 1, b \geq 1, \gcd(a, b) =$ largest integer that divides both a and b.

Example: $\gcd(75{,}45) =$

Common divisors: $1, 3, 5, 15$

$\gcd(a, a) = a$

How do we find gcd of large numbers?

$$a = 371\,435\,805 = \overbrace{3^2 \cdot 5^1 \cdot 13^4 \cdot 17^2}^{\text{Prime Factorization}}$$
$$b = 137\,916\,675 = 3^4 \cdot 5^2 \cdot 13^3 \cdot 31$$

$$\gcd(a, b) =$$

Euclid’s Algorithm

Greatest common divisor.

$a \geq 1, b \geq 1, \gcd(a, b) =$ largest integer that divides both a and b.

Example: $\gcd(75,45) =$

Common divisors: $1, 3, 5, 15$

$\gcd(a, a) = a$

How do we find gcd of large numbers?

$$a = 371\,435\,805 = \overbrace{3^2 \cdot 5^1 \cdot 13^4 \cdot 17^2}^{\text{Prime Factorization}}$$

$$b = 137\,916\,675 = 3^4 \cdot 5^2 \cdot 13^3 \cdot 31$$

$$\gcd(a, b) = 3^2 \cdot 5^1 \cdot 13^3 = 98\,865$$

Euclid's Algorithm

Greatest common divisor.

$a \geq 1, b \geq 1, \gcd(a, b) =$ largest integer that divides both a and b.

Example: $\gcd(75{,}45) =$

Common divisors: $1, 3, 5, 15$

$\gcd(a, a) = a$

How do we find gcd of large numbers?

$$a = 371\,435\,805 = \overbrace{3^2 \cdot 5^1 \cdot 13^4 \cdot 17^2}^{\text{Prime Factorization}}$$
$$b = 137\,916\,675 = 3^4 \cdot 5^2 \cdot 13^3 \cdot 31$$

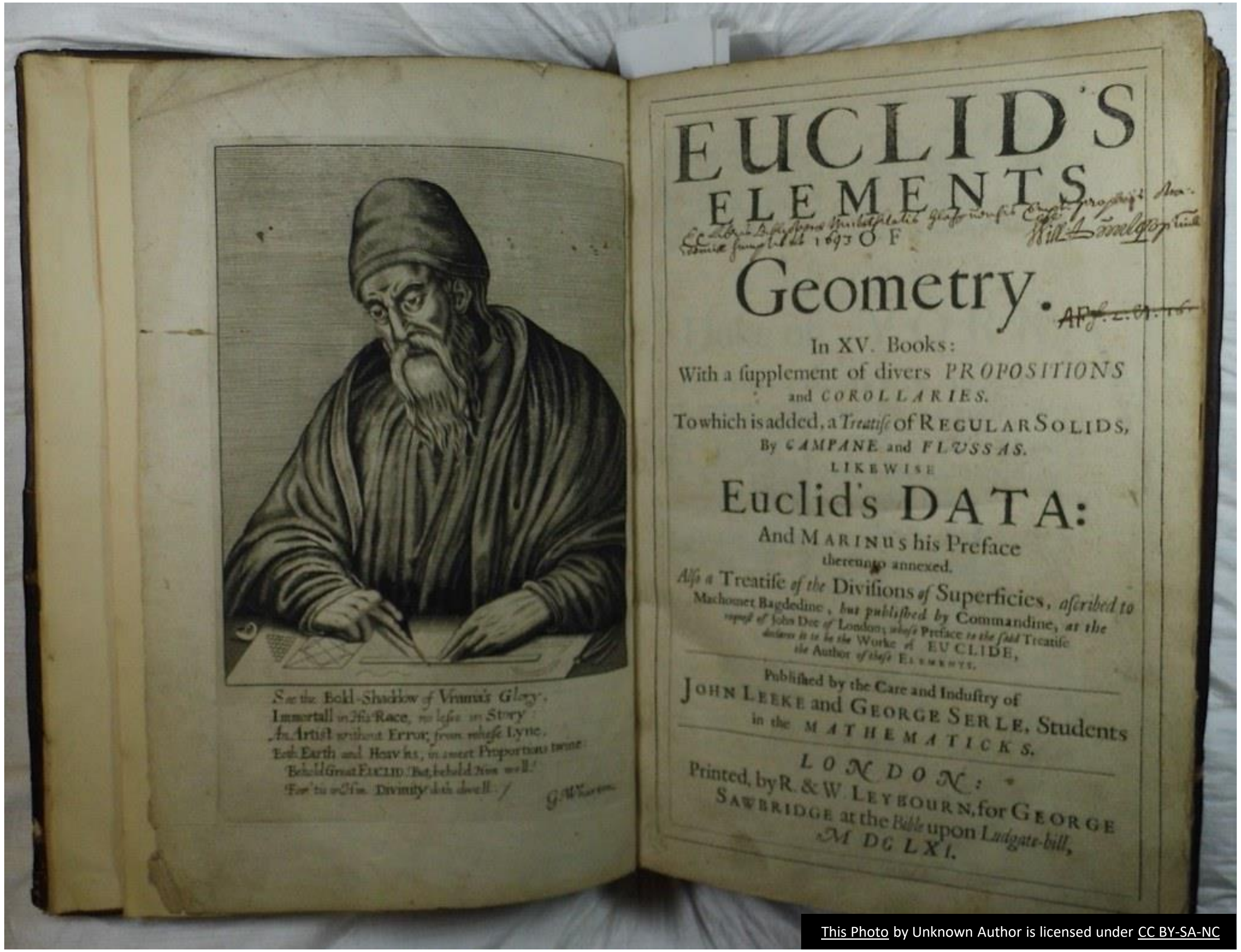
$$\gcd(a, b) = 3^2 \cdot 5^1 \cdot 13^3 = 98\,865$$

Compute Prime Factorization of a and b

Very slow!

Much computer security is based on the fact that prime factorization is very slow

An integer with 1000 digits will take 1000's of years to compute PF.



Euclid's Algorithm

Greatest common divisor.

$a \geq 1, b \geq 1, \gcd(a, b) =$ largest integer that divides both a and b.

Easy, fast algorithm to compute $\gcd(a, b)$.

Invented by Euclid around 300 BC.

Uses the Modulo operation.

Modulo

Modulo operation:

a mod b = remainder of a divided by b

$a = qb + r,$

$0 \leq r < b, q \geq 0$

$q =$ quotient
$r =$ remainder

$a \bmod b = r$

$17 \bmod 5 \ = \ 2$

$17 \bmod 17 \ = 0$

$17 \bmod 1 \ = 0$

$17 \bmod 19 \ = 17$

$a = qb + r$

$17 = \ 3 \cdot 5 + 2$

$17 = 1 \cdot 17 + 0$

$17 = 17 \cdot 1 + 0$

$17 = 19 \cdot 0 + 17$

Euclid Algorithm

$a \bmod b = r$

$a = qb + r,$

$0 \leq r < b, q \geq 0$

Algorithm Euclid(a, b): $// \, a \geq b \geq 1$

  $r = a \bmod b$

  if $r = 0$: return b

  if $r \geq 1$:

    return Euclid(b, r) $// \, b \geq r \geq 1$

Euclid(75, 45):

Using prime factorization:

$$75 = 3 \cdot 5^2$$
$$45 = 3^2 \cdot 5$$

$$\gcd(75{,}45) = 3 \cdot 5 = 15$$

Euclid Algorithm

$a \bmod b = r$

$a = qb + r,$

$0 \le r < b, q \ge 0$

Algorithm Euclid(a, b): $// \; a \ge b \ge 1$
  $r = a \bmod b$
  if $r = 0$: return b
  if $r \ge 1$:
    return Euclid(b, r) $// \; b \ge r \ge 1$

$\mathrm{gcd}(75,45) = 15$

Euclid(75, 45):
  $r = 75 \bmod 45 = 30$
  Euclid(45, 30)
    $r = 45 \bmod 30 = 15$
      Euclid(30, 15)
      $r = 30 \bmod 15 = 0$
      return 15

It is correct for this input.

Euclid Algorithm

$a \bmod b = r$

$a = qb + r,$

$0 \le r < b, q \ge 0$

Algorithm Euclid(a, b): $// a \ge b \ge 1$
  $r = a \bmod b$
  if $r = 0$: return b
  if $r \ge 1$:
    return Euclid(b, r) $// b \ge r \ge 1$

$\gcd(75{,}45) = 15$

We have to argue Euclid is correct $\forall a, b$ and also that Euclid terminates.

Euclid(a,b) is correct if it returns gcd(a,b) in both the base case and the recursive case.

Thus we must prove:

1. When $r = 0$, gcd(a,b) = b
2. When $r \ge 1$,
   a. Euclid(b,r) returns gcd(b,r) **and**
   b. gcd(b,r) = gcd(a,b).

We will start by proving 1 and 2b, then we prove 2a after.

Euclid Algorithm

Lemma 1: $a \geq b \geq 1, r = a \bmod b$

a) if $r = 0$ then $\gcd(a, b) = b$

b) if $r \geq 1$ then $\gcd(a, b) = \gcd(b, r)$

$a = qb + r$

a) if $r = 0$,

$\gcd(a, b) = \gcd(qb, b) = b$,

so a) is true

b) if $r \geq 1$ then $\gcd(a, b) = \gcd(b, r)$ is true if

all common divisors of a and b = all common divisors of b and r.

To argue this we must show a bijection between all common divisors of a and b and all common divisors of b and r.

i) First we show that if d is a common divisor of a and b then d is also a common divisor of b and r.
ii) Second we show that if d is a common divisor of b and r then d is also a common divisor of a and b.

Euclid Algorithm

Lemma 1: $a \geq b \geq 1, r = a \bmod b$

a) if $r = 0$ then $\gcd(a, b) = b$
$\boxed{\text{b) if } r \geq 1 \text{ then } \gcd(a, b) = \gcd(b, r)}$

$a = qb + r$

To show: i) if d is a common divisor of a and b then d is also a common divisor of b and r.

$a = qb + r$

$r = a - qb$

a is a multiple of d, that is $a = da'$
b is a multiple of d, that is $b = db'$

If both a and b are multiples of d, then we can write r as:

$$r = d(a' - qb')$$

and r is also a multiple of d.

Now we must argue the other direction

Euclid Algorithm

Lemma 1: $a \geq b \geq 1, r = a \bmod b$

a) if $r = 0$ then $\gcd(a, b) = b$
b) if $r \geq 1$ then $\gcd(a, b) = \gcd(b, r)$

$a = qb + r$

To show: i) if d is a common divisor of a and b then d is also a common divisor of b and r.

ii) if d is a common divisor of b and r then d is also a common divisor of a and b.

$a = qb + r$

If b is a multiple of d, then $b = db'$.
If r is a multiple of d, then $r = dr'$.

Thus we can rewrite a as:

$$a = d(qb' + r')$$

and a is a multiple of d.

Euclid Algorithm

Lemma 1: $a \geq b \geq 1, r = a \bmod b$

a) if $r = 0$ then $\gcd(a, b) = b$
b) if $r \geq 1$ then $\gcd(a, b) = \gcd(b, r)$

$a = qb + r$

To show: i) if d is a common divisor of a and b then d is also a common divisor of b and r.

ii) if d is a common divisor of b and r then d is also a common divisor of a and b.

If d is a common divisor of a and b then d is also a common divisor of b and r → True .

If d is a common divisor of b and r then d is also a common divisor of a and b → True.

Therefore all common divisors are the same.

Therefore it must be that $\gcd(a, b) = \gcd(b, r)$. Thus b) is true.

So Lemma 1 is True.

Euclid Algorithm

$a \bmod b = r$

$a = qb + r, 0 \leq r < b, q \geq 0$

Algorithm Euclid(a, b): // $a \geq b \geq 1$
  $r = a \bmod b$
  if $r = 0$: return b
  if $r \geq 1$:
    return Euclid(b, r) // $b \geq r \geq 1$

Lemma 1:
a) if $r = 0$ then $\gcd(a, b) = b$
b) if $r \geq 1$ then $\gcd(a, b) = \gcd(b, r)$

Now we can prove that Euclid is correct using induction. To successfully use induction we require that Euclid terminates, which means we can rank the calls to Euclid.

Can do induction on size of a.

Base case: $a = 1$ (which implies that $b = 1$):

Euclid(1,1) returns 1 which is true.

Inductive hypothesis: Assume that Euclid(b, r) terminates when $b < a$.

Since $b < a$, Euclid terminates.

Euclid Algorithm

a mod $b = r$

$a = qb + r, 0 \leq r < b, q \geq 0$

Algorithm Euclid(a, b): $// \, a \geq b \geq 1$

$r = a$ mod b

if $r = 0$: return b

if $r \geq 1$:

return Euclid(b, r) $// \, b \geq r \geq 1$

Lemma 1:

a) if $r = 0$ then $\gcd(a, b) = b$

b) if $r \geq 1$ then $\gcd(a, b) = \gcd(b, r)$

Euclid(a,b):

if $r = 0$

return b (which is correct)

else

return Euclid(b, r), where $b < a$

Because $b < a$, Euclid(b, r) is assumed correct by induction

-that is, it returns $\gcd(b, r)$.

Since gcd(a, b) = gcd(b, r), Euclid(a, b) is correct.

Euclid Algorithm

$a \bmod b = r$

$a = qb + r, 0 \le r < b, q \ge 0$

Algorithm Euclid(a, b): // $a \ge b \ge 1$
  $r = a \bmod b$*
  if $r = 0$: return b
  if $r \ge 1$:
    return Euclid(b, r) // $b \ge r \ge 1$

Lemma 1:
a) if $r = 0$ then $\gcd(a, b) = b$
b) if $r \ge 1$ then $\gcd(a, b) = \gcd(b, r)$

Euclid is correct. Let's analyze the runtime.

How efficient is Euclid(a,b)? Start with the example Euclid(75, 45):

$M(a, b)$ = no. of times * (mod) is executed.

Euclid(75, 45):
  $r = 75 \bmod 45 = 30$*
  Euclid(45, 30)
    $r = 45 \bmod 30 = 15$*
      Euclid(30, 15)
      $r = 30 \bmod 15 = 0$*
      return 15

$M(75,45) = 3$

Euclid Algorithm

$a \bmod b = r$

$a = qb + r, 0 \le r < b, q \ge 0$

Algorithm Euclid(a, b): $// \, a \ge b \ge 1$
  $r = a \bmod b$
  if $r = 0$: return b
  if $r \ge 1$:
    return Euclid(b, r) $// \, b \ge r \ge 1$

Lemma 1:
a) if $r = 0$ then $\gcd(a, b) = b$
b) if $r \ge 1$ then $\gcd(a, b) = \gcd(b, r)$

How efficient is Euclid(a,b)?

$M(a, b)$ = number of times line * is executed.

Start with easy analysis:

Euclid(a, b):
  always $b \ge 1$
  decreases by ≥ 1

$M(a, b) \le b$

But we can do a better analysis based on the Fibonnacci sequence.

Euclid Algorithm

$a \bmod b = r$

$a = qb + r, 0 \leq r < b, q \geq 0$

Algorithm Euclid(a, b): $// a \geq b \geq 1$

$r = a \bmod b$

if $r = 0$: return b

if $r \geq 1$:

return Euclid(b, r) $// b \geq r \geq 1$

Lemma 1:

a) if $r = 0$ then $\gcd(a, b) = b$

b) if $r \geq 1$ then $\gcd(a, b) = \gcd(b, r)$

How efficient is Euclid(a,b)?

$M(a, b) =$ number of times line * is executed.

Euclid(75, 45):

$r = 75 \bmod 45 = 30*$

Euclid(45, 30)

$r = 45 \bmod 30 = 15*$

Euclid(30, 15)

$r = 30 \bmod 15 = 0*$

return 15

Look at the sequence of arguments in reverse:

$$75, 45, 30, 15$$

Euclid Algorithm

$a \bmod b = r, \quad a = qb + r$

Algorithm Euclid(a, b): // $a \geq b \geq 1$
  $r = a \bmod b$ *
  if $r = 0$: return b
  if $r \geq 1$:
    return Euclid(b, r) // $b \geq r \geq 1$

$M(a, b) =$ number of times line * is executed.

Fibonacci: $f_0 = 0, f_1 = 1, f_2 = 1, f_3 = 2$, etc

Lemma 2: $a \geq b \geq 1, m = M(a, b)$
Then $a \geq f_{m+1}, b \geq f_m$

Euclid Algorithm

$a \bmod b = r, \quad a = qb + r$

Algorithm Euclid(a, b): $// a \geq b \geq 1$
  $r = a \bmod b$ *
  if $r = 0$: return b
  if $r \geq 1$:
    return Euclid(b, r) $// b \geq r \geq 1$

$M(a, b)$ = number of times line * is executed.

Fibonacci: $f_0 = 0, f_1 = 1, f_2 = 1, f_3 = 2$, etc

Lemma 2: $a \geq b \geq 1, m = M(a, b)$
Then $a \geq f_{m+1}, b \geq f_m$

The idea behind it is this: $a \geq b + r$

Which means if we look at all the values that we use in calls to Euclid(a, b), they grow like the Fibonacci sequence.

Euclid Algorithm

a mod $b = r, \quad a = qb + r$

Algorithm Euclid(a, b): $// \, a \geq b \geq 1$
  $r = a$ mod b *
  if $r = 0$: return b
  if $r \geq 1$:
    return Euclid(b, r) $// \, b \geq r \geq 1$

$M(a, b) =$ number of times line * is executed.

Fibonacci: $f_0 = 0, f_1 = 1, f_2 = 1, f_3 = 2$, etc

Lemma 2: $a \geq b \geq 1, m = M(a, b)$
Then $a \geq f_{m+1}, b \geq f_m$

The idea behind it is this: $a \geq b + r$

Which means if we look at all the values that we use in calls to Euclid(a, b), they grow like the Fibonacci sequence.

$$a \geq b + r$$
$$f_n = f_{n-1} + f_{n-2}$$

So if $r \geq f_{n-2}$ and $b \geq f_{n-1}$ then $a \geq f_n$.
Then the numbers in Euclid(a, b) grow *at least* as fast as the Fibonacci sequence.

Euclid Algorithm

$a \bmod b = r, \quad a = qb + r$

Algorithm Euclid(a, b): $// \, a \geq b \geq 1$
  $r = a \bmod b$ *
  if $r = 0$: return b
  if $r \geq 1$:
    return Euclid(b, r) $// \, b \geq r \geq 1$

$M(a, b) =$ number of times line * is executed.

Fibonacci: $f_0 = 0, f_1 = 1, f_2 = 1, f_3 = 2$, etc

Lemma 2: $a \geq b \geq 1, m = M(a, b)$
Then $a \geq f_{m+1}, b \geq f_m$
Induction on m: Base case $m = 1$, no recursive call, $r = a \bmod b = 0$, so we have 1 mod call which is equal to $f_1 = 1$.

$b \geq 1 = f_1$ which is true
$a \geq b \geq 1 = f_2$ which is true

Inductive Step: $m \geq 2$
Euclid(a,b):
  $r = a \bmod b \geq 1$
  Euclid(b, r)
  …$m - 1$ *recursive calls in total…*

Euclid Algorithm

$a \bmod b = r, \quad a = qb + r$

Algorithm Euclid(a, b): $// \, a \geq b \geq 1$
  $r = a \bmod b$ *
  if $r = 0$: return b
  if $r \geq 1$:
    return Euclid(b, r) $// \, b \geq r \geq 1$

$M(a, b) =$ number of times line * is executed.

Fibonacci: $f_0 = 0, f_1 = 1, f_2 = 1, f_3 = 2$, etc

Lemma 2: $a \geq b \geq 1, m = M(a, b)$
Then $a \geq f_{m+1}, b \geq f_m$

Inductive Step: $m \geq 2$

Assume that for any call to Euclid(b, r) where $M(b, r) = m - 1$ that

$b \geq f_m, r \geq f_{m-1}$.

$$\begin{aligned} a &= qb + r \\ &\geq b + r \\ &\geq f_m + f_{m-1} \\ &= f_{m+1} \end{aligned}$$

Euclid Algorithm

$a \bmod b = r, \quad a = qb + r$

Algorithm Euclid(a, b): $// \; a \geq b \geq 1$

$r = a \bmod b$ *

if $r = 0$: return b

if $r \geq 1$:

return Euclid(b, r) $// \; b \geq r \geq 1$

$M(a, b)$ = number of times line * is executed.

Fibonacci: $f_0 = 0, f_1 = 1, f_2 = 1, f_3 = 2$, etc

Lemma2: $a \geq b \geq 1, m = M(a, b)$

Then $a \geq f_{m+1}, b \geq f_m$ → True

Lemma 3: $a \geq b \geq 1, M(a, b) \leq 1 + \log_\phi b$

$$\phi = \frac{1 + \sqrt{5}}{2}$$

if $a = b, M(a, b) = 1 \leq 1 + \log_\phi b$

if $a > b, M(a, b) = m$

$b \geq f_{m+1} \geq \phi^{m-1}$ (exercise using $\phi^2 = \phi + 1$)

$$\log_\phi b \geq m - 1$$

$$m \leq 1 + \log_\phi b$$