What have we covered (recently)?

counting operations

runtime

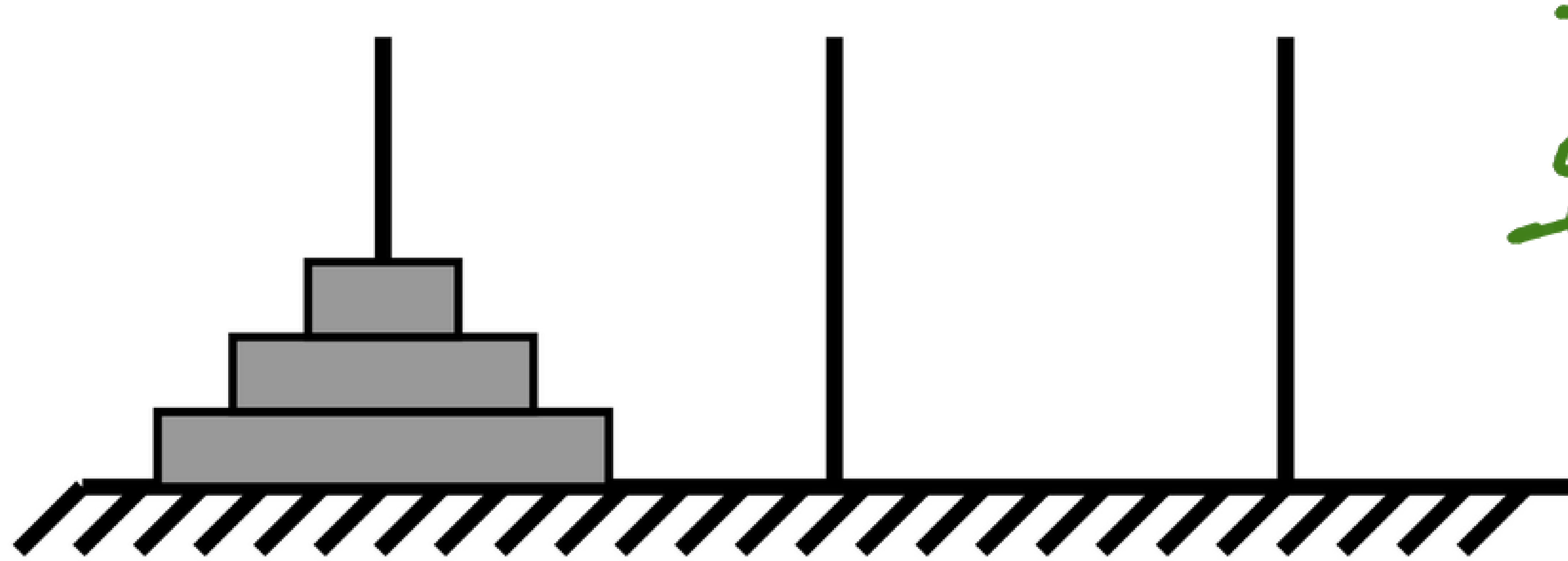
big-O notation

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

(arithmetic) $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$

Goals for today:

- Develop recurrence relations to analyze the performance of recursive algorithms.
- Solve recurrence relations using the *guess and check* method.
- Solve recurrence relations using the *expand and pray* method.



Tower of Hanoi

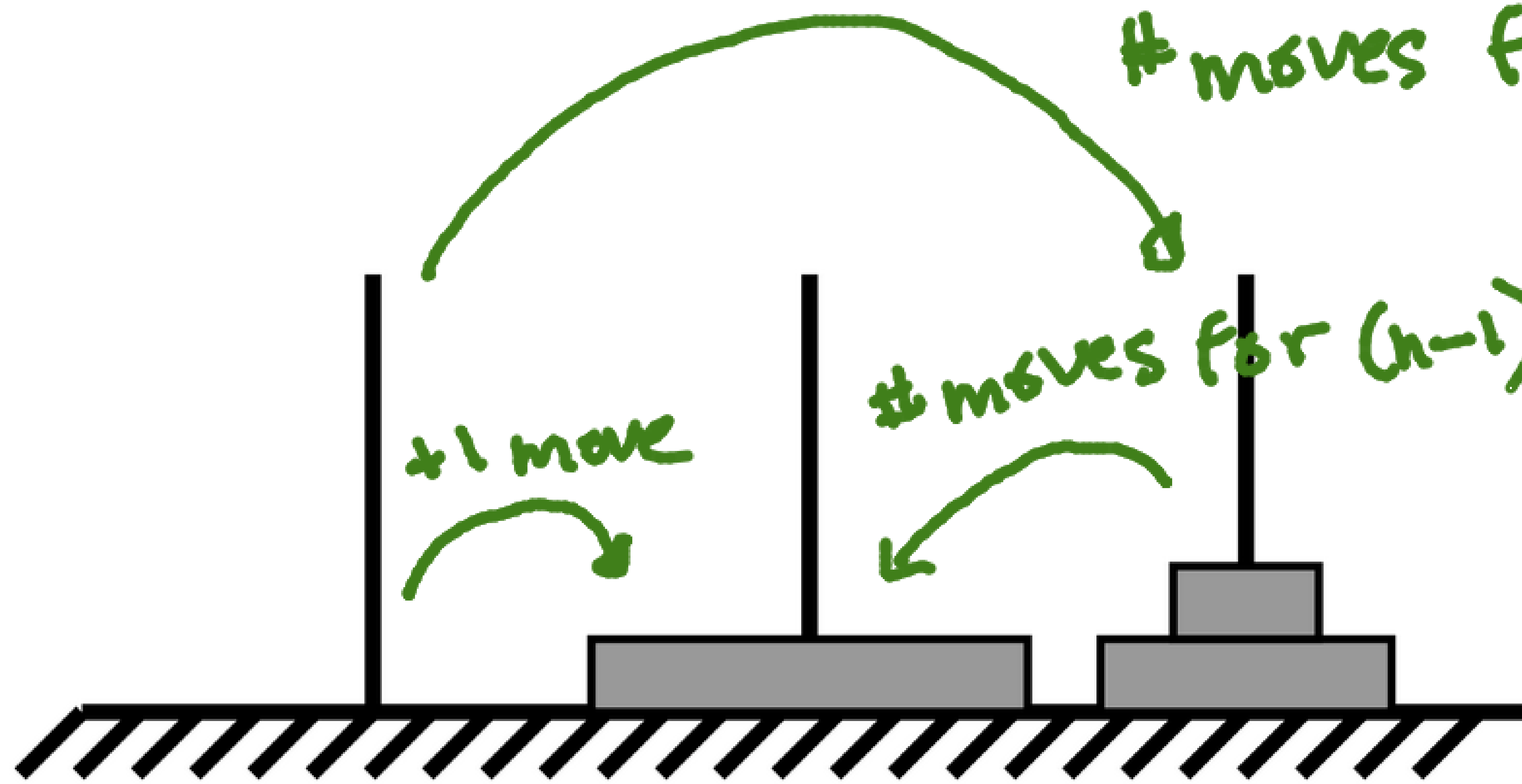
goal: move stack of n disks from one rod to another (count # moves)

Rules:

1. Move one disk at a time.
2. Every move displaces a disk from the top of one stack to the top of another (or empty rod).
3. No larger disk can ever be placed on top of a smaller one.

$n=1$
$n=2$
$n=3$

Counting the number of moves in the Tower of Hanoi problem.



let $T(n)$ represent # moves for a n-disk stack.

n	# moves
1	1
2	3
3	7
4	15
⋮	⋮

moves $= 2^n - 1$
(we'll come back to this, check with induction)

$$\boxed{T(n) = 2T(n-1) + 1}$$

recurrence relation

base case: $\boxed{T(1) = 1}$

After guessing, check with induction!

guess $T(n) = 2^n - 1$

We use a proof by induction on n. Let the induction hypothesis be the predicate $p(n)$: " $T(n) = 2^n - 1$ solves the recurrence relation $T(n) = 2T(n-1)+1$ with base case $T(1) = 1$ "

... skipping steps

Inductive step: Assume $p(n)$ is true, i.e. $T(n) = 2^n - 1$ solves the recurrence relation $T(n) = 2T(n-1) + 1$ with $T(1) = 1$. We will prove $p(n+1)$ is true. Starting with the recurrence for an input of $n+1$, we have:

$$
\begin{aligned}
T(n+1) &= 2T(n+1-1) + 1 \\
&= 2T(n) + 1 \\
&= 2(2^n - 1) + 1 \\
&= 2^{n+1} - 1
\end{aligned}
$$

$T(n) = 2^n - 1$ by assumption that $p(n)$ is true.

verifies $p(n+1)$ is true

$p(n) = \#\text{ways} = m(m-1)^{n-1}$

$\cancel{(m-1)p(n)}$

A slightly better way: expand recurrence relation, and hope we see a pattern.

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

plug in (expand) $T(n-1) = 1 + 2T(n-2)$

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

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

$T(n-2) = 1 + 2T(n-3)$

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

$$= 1 + 2 + 2^2(1 + 2T(n-3))$$

$T(n-3) = 1 + 2T(n-4)$

$$= 1 + 2 + 2^2 + 2^3 T(n-3)$$

$$= 1 + 2 + 2^2 + 2^3(1 + 2T(n-4))$$

$$= 1 + 2 + 2^2 + 2^3 + 2^4 T(n-4)$$

$$= 1 + 2 + 2^2 + 2^3 + \dots + 2^{i-1} + 2^i T(\underbrace{n-i})$$

when do we stop expanding?

$T(1) = 1 \quad n - i = 1$

$i = n-1$

$r = 2$

$$= \sum_{i=0}^{\boxed{n-1}} 2^i = \frac{1 - r^{\boxed{n-1}+1}}{1-r} = \frac{1-2^n}{1-2} = \boxed{2^n - 1}$$

☺

Types of recurrence relations we will see.

linear recurrence relations (Wednesday)

```python
def fib(n: int) -> int:
    if n <=1:
      return n
    return fib(n - 1) + fib(n - 2)
```

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

divide-and-conquer recurrences (Friday)

```python
def binary_search(a: list[int], value: int) -> bool:
  n = len(a)
  if n == 0:
    return False
  elif n == 1:
    if a[0] == value:
      return True
    return False

  m = n // 2
  if a[m] <= value:
    return binary_search(a[m:], value)
  else:
    return binary_search(a[:m], value)
```