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.

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

Let $T(n)$ represent the number of moves for a $n$-disk stack.

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

base case: $T(1) = 1$

$\text{# moves} = 2^n - 1$

We'll come back to this; check with induction.

$n$ | # moves
--- | ---
1 | 1
2 | 3
3 | 7
4 | 15

$\text{# moves for (n-1) disk stack}$

$\text{# moves for (n-1) disk stack}$
After guessing, check with induction!

We use a proof by induction on $n$. Let the induction hypothesis be the predicate $p(n)$:

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

**Inductive step:** Assume $p(n)$ is \textit{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 \textit{true}. Starting with the recurrence for an input of $n + 1$, we have:

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

verifies $p(n+1)$ is \textit{true}.

$T(n) = 2^n - 1$ by assumption that $p(n)$ is \textit{true}.
A slightly better way: expand recurrence relation, and hope we see a pattern.

\[ T(n) = 2T(n-1) + 1 \quad \text{plug in (expand)} \quad T(n-1) = 1 + 2T(n-2) \]

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

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

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

\[ = 1 + 2 + 2^2\left(1 + 2T(n-3)\right) \]

\[ = 1 + 2 + 2^2 + 2^3T(n-3) \]

\[ = 1 + 2 + 2^2 + 2^3\left(1 + 2T(n-4)\right) \]

\[ = 1 + 2 + 2^2 + 2^3 + 2^4T(n-4) \]

\[ = 1 + 2 + 2^2 + 2^3 + \ldots + 2^{i-1} + 2^iT(n-i) \]

\[ \sum_{i=0}^{n-1} 2^i = \frac{1 - r^{n+1}}{1 - r} = \frac{1 - 2^{n}}{1 - 2} = \sqrt{2^n - 1} \]

\[ \text{when do we stop expanding?} \quad T(1) = 1 \quad n - i = 1 \quad i = n - 1 \]

\[ \]
Types of recurrence relations we will see.

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

```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)
```