Today:
- recursion algorithms
- self reduction
- analysis of runtime through
  recursion
- merge sort

Reduction

\[
NFA \rightarrow DFA \rightarrow RE
\]
follow A reduces to problem B
iff we can deduce a solution for B
and transform it to a solution for A.

\[
L = A \rightarrow \text{pe}
\]
transformed solution

\[
NFA \rightarrow B \rightarrow \text{pe}
\]

"Self reduction" reducing to a smaller instance of
the same problem.
- design
- correctness by induction these case

- runtime \( T(n) = S(T(n-1)) \)
  analyze with recurrence

Towers of Hanoi

Goal

Moves:

Rule: can only take a tile from top,
and can only place an empty peg
of a peg larger hole

Base Case:

\[
n = 0 \quad \checkmark \quad T(0) = 0
\]

\[
n = 1 \quad \checkmark \quad T(1) = 1
\]

\[
n = 2 \quad \checkmark \quad T(2) = 3
\]
How to generalize:
- Assume I have $H(n')$ is a solution
  to move $n'$ disks to an empty peg.
  To prove $H(n)$ has a solution for $n=n'+1$

$\begin{array}{c}
n-1 \\
\text{ } \\
n \end{array}
\xrightarrow{\text{move left disk to middle peg}}
\begin{array}{c}
n-1 \\
\text{ } \\
n \end{array}
\Rightarrow
\begin{array}{c}
n-1 \\
\text{ } \\
n \end{array}
\Rightarrow
\begin{array}{c}
n-1 \\
\text{ } \\
n \end{array}
$

- Moving the right peg $\Rightarrow$ can move to center peg too.

- Adding a large tile at bottom doesn't affect moves.

0. Use $H(n-1)$ to move $n-1$ tiles to middle peg.
1. Move $n$ tile to right peg.
2. Use $H(n-1)$ to move $n-1$ tiles to middle peg.

$\Rightarrow
\begin{array}{c}
n-1 \\
\text{ } \\
n \end{array}
$

---

$T(n) = T(n-1) + 1 + T(n-1)$
$= 2T(n-1) + 1$
$T(0) = 0$

- Guess and verify by induction

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

  Base: $T(0) = 2^0 - 1 = 0$

  Induction:
  $T(n) = 2T(n-1) + 1$
  $= 2(2^{n-1} - 1) + 1$
  $= (2^n - 2) + 1$
  $= 2^n - 1$

---

**Merge Sort:**

$\begin{array}{c}
3 \\
7 \\
8 \\
1 \\
\end{array}
\begin{array}{c}
2 \\
4 \\
0 \\
9 \\
\end{array}$

$2T(n) + T(n) + 2 + 2$
Analysis:
- Each layer is in comparison.
- How many layers? \( \log_2 n \)

\[
T(n) = \frac{n \left( \log_2 n + 1 \right)}{2} + T(n/2) + \frac{n}{2} \log_2 \frac{n}{2} + n
\]

- \( T(1) = 0 \)
- \( T(2) = 1 \)

IH \( T(n/2) = \frac{n}{2} \left( \log_2 n + 1 \right) \)

\[
T(n) = 2 \left( \frac{n}{2} \left( \log_2 \frac{n}{2} + 1 \right) \right) + n \log_2 n/2 + n + n
\]

\[
= n \log_2 n - n \log_2 n + n + n
\]

Asymptotic analysis:

- \( T(n) = n \log n + n \)

\( T(n) \in O(n \log n) \)

Asymptotic for some constant factor stack, only has to hold for large \( n \).

\( T(n) = O(S(n)) \)
\[ T(n) = \Theta(F(n)) \]

- **worst case**
- **upper bound**
- **lower bound**

Model of computation:
- primitive math operations \( \text{add(1)step} \)
- comparisons \( \text{1 step} \)
- array accesses \( \text{1 step} \)

Why do asymptotic analysis?

- existing \( O(n^2) \)
- my goal \( O(n^2) \)