CSCI 3100: Algorithms

Today:
Now: Recursion

- Induction started at the bottom and builds up.

Recursion: the natural dual idea:
Recurrence relations:

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

\[ M(n) = 2M\left(\frac{n}{2}\right) + n \]

\[ T(n) = T\left(\frac{3n}{4}\right) + n \]

How to solve?
Master Thm:

Let \( T(n) = \sum_{d \mid n} a_T(n/d) + nc \) if \( n > 1 \)

for constants \( a, b, c, d \)

Then: - if \( \log_b a < c \), \( T(n) = \Theta(f(n)) \)
- if \( \log_b a = c \), \( T(n) = \Theta(n^c \log n) \)
- if \( \log_b a > c \), \( T(n) = \Theta(n^{\log_b a}) \)
There are other versions:

**The Master Theorem.** The recurrence $T(n) = aT(n/b) + f(n)$ can be solved as follows.

- If $a f(n/b) = \kappa f(n)$ for some constant $\kappa < 1$, then $T(n) = \Theta(f(n))$.
- If $a f(n/b) = K f(n)$ for some constant $K > 1$, then $T(n) = \Theta(n^{\log_b a})$.
- If $a f(n/b) = f(n)$, then $T(n) = \Theta(f(n) \log_b n)$.
- If none of these three cases apply, you’re on your own.

Really, saying the same thing: (We’ll talk about the proof later...)
Recursive algorithms:
Based on reduction:
Reduce to a smaller instance of the same problem.

Necessary pieces (like induction):
Classical example: Towers of Hanoi
Strategy: think recursively!

Start small:

[Diagram of a simple structure with a line and some small shapes]
Bigger picture:

\[
\text{HANOI}(n, \text{src}, \text{dst}, \text{tmp}):
\]
\[
\text{if } n > 0
\]
\[
\text{HANOI}(n - 1, \text{src}, \text{tmp}, \text{dst})
\]
\[
\text{move disk } n \text{ from src to dst}
\]
\[
\text{HANOI}(n - 1, \text{tmp}, \text{dst}, \text{src})
\]

The Tower of Hanoi algorithm; ignore everything but the bottom disk
Proof of correctness:
Runtime:
Next: Sorting & searching

Interview question: Name the best sorting algorithm & justify your answer.
Merge Sort:
(supposedly, suggested by von Neumann in 1945)

Idea:
1. 
2. 
3. 
In action:

| Input:   | S O R T I N G E X A M P L   |
| Divide:  | S O R T I N G E X A M P L   |
| Recurse: | I N O S R T A E G L M P X  |
| Merge:   | A E G I L M N O P R S T X  |

A mergesort example.
Algorithm: merging is the only hard part!

**MergeSort**$(A[1..n])$:
if $n > 1$
  $m \leftarrow \lfloor n/2 \rfloor$
  **MergeSort**$(A[1..m])$
  **MergeSort**$(A[m+1..n])$
  **Merge**$(A[1..n], m)$

**Merge**$(A[1..n], m)$:
  $i \leftarrow 1; j \leftarrow m + 1$
  for $k \leftarrow 1$ to $n$
    if $j > n$
      $B[k] \leftarrow A[i]; i \leftarrow i + 1$
    else if $i > m$
      $B[k] \leftarrow A[j]; j \leftarrow j + 1$
    else if $A[i] < A[j]$
      $B[k] \leftarrow A[i]; i \leftarrow i + 1$
    else
      $B[k] \leftarrow A[j]; j \leftarrow j + 1$
  for $k \leftarrow 1$ to $n$
  $A[k] \leftarrow B[k]$
Correctness:
Runtime: