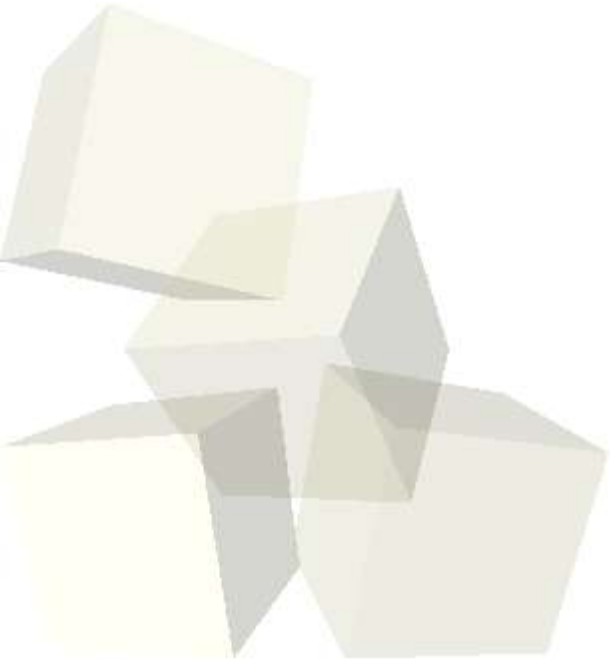
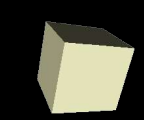


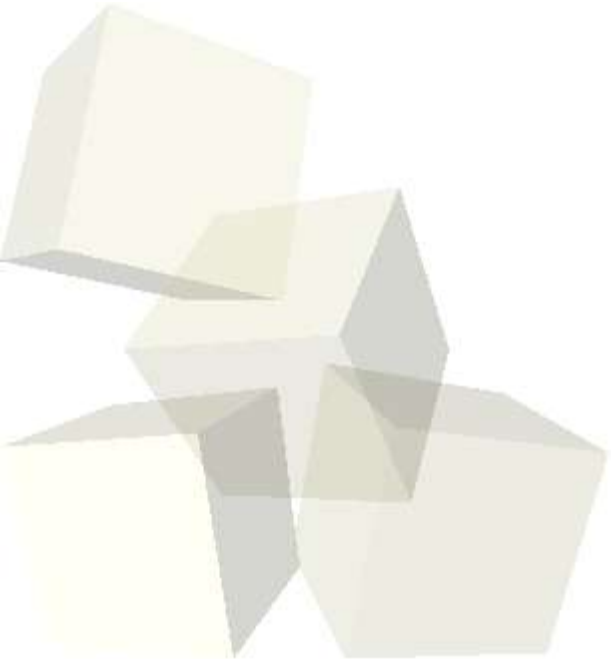
Recurrences

Alexandre David



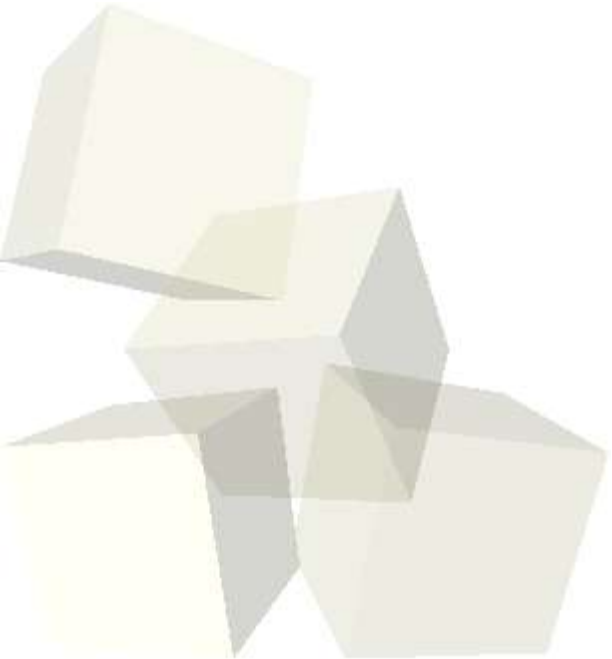
Recurrences

- A **recurrence** is an equation or inequality that describes a function in terms of its value on smaller inputs.
- Methods for solving recurrences:
  - Substitution method
  - Recursion-tree method
  - Master method



The Substitution Method

- Two steps:
  - Guess the form of the solution
  - Use induction to find the constants and prove that the solution works
- Problem: come up with a good guess
  - Use recursion trees
  - Correct the guess



The Recursion-Tree Method

- Each node represents the cost of a single subproblem.
- Useful when it describes the running time of a divide-and-conquer algorithm.
- Used to generate a good guess or as a direct proof of a solution to a recurrence.
- Example: $T(n)=3T(n/4)+\theta(n^2)$
  - Construct recursion tree to obtain a guess
  - Use the substitution method for the proof

The Master Method

- For recurrences of the form $T(n)=aT(n/b)+f(n)$ where $a \geq 1$ and $b>1$ are constants and $f(n)$ is asymptotically positive.
- Theorem: $T(n)$ can be bounded as follows
  - if $f(n)=O(n^{\log_b a-\epsilon})$ for some $\varepsilon>0$ then $T(n)=\Theta(n^{\log_b a})$
  - if $f(n)=\Theta(n^{\log_b a})$ then $T(n)=\Theta(n^{\log_b a} lg\, n)=\Theta(f(n) lg\, n)$
  - if $f(n)=\Omega(n^{\log_b a+\epsilon})$ for some $\varepsilon>0$ and if $af(n/b) \leq cf(n)$ for some $c<1$ then $T(n)=\Theta(f(n))$

- Intuition: compare f(n) to $n^{\log_b a}$, the larger is the solution. Smaller: polynomially smaller by a factor of n^{ϵ}. Larger: polynomially larger + “regularity” condition.

Master Method: Examples

- $T(n)=9T(n/3)+n$

$a=9,\ b=3,\ f(n)=n$ $\quad n^{\log_b a}=n^{\log_3 9}=n^2$

Case 1 with $\varepsilon=1$, we conclude $T(n)=\Theta(n^2)$.

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

$a=1,\ b=2/3,\ f(n)=1$ $\quad n^{\log_b a}=n^{\log_{3/2} 1}=n^0=1$

Case 2, we conclude $T(n)=\Theta(\lg n)$.

- $T(n)=3T(n/4)+n\lg n$

$a=3,\ b=4,\ f(n)=n\lg n$ $\quad n^{\log_b a}=n^{\log_4 3}=O\left(n^{0.793}\right)$

$f(n)=\Omega\left(n^{\log_b a+\epsilon}\right)$

Case 3, check regularity:

$af(n/b)=3(n/4)\lg(n/4)\le(3/4)n\lg n=cf(n)$.

We conclude $T(n)=\Theta(n\lg n)$.

Master Method: Example

- $T(n)=2T(n/2)+n\lg n$

$a=2$, $b=2$, $f(n)=n\lg n$, $n^{\log_b a}=n$

Case 3?

$f(n)=n\lg n$ is not polynomially larger than n: no $\varepsilon>0$ such that $n\lg n=\Omega(n^{1+\varepsilon})$.

We cannot apply the master theorem.

Master Theorem: Proof Idea

- Proof for a sub-domain: $n=1,b,b^2,\ldots$
  - compute the cost with a recursion tree (lemma 4.2)
    leaves: $\Theta\left(n^{\log_b a}\right)$
    +tree: $\sum_{j=0}^{\log_b n-1} a^j f\left(n/b^j\right)$
  - bound the 2nd term with 3 cases (as in the theorem) (lemma 4.3)
  - evaluate the sum (asymptotically) using lemma 4.3.
- Extend the proof for any n.