Math 135 - Divide & Conquer Recurrences

Note Title 10/31/2012

Divide and Conquer Recurrences

Non-Linear, but in terms of smaller values in the sequence based on division; eg: $f(n) = a f(\frac{n}{b}) + g(n)$

Ex:

$B(n) = B(\frac{n}{2}) + 1$ ← $O(1)$

$M(n) = 2M(\frac{n}{2}) + n$

$f(n) = 7f(\frac{n}{2}) + \frac{15n^2}{4}$

Why do we care?

Ex: Binary Search $(x, a_1, \ldots, a_n)$:

$O(1)$ [
- Calculate $n/2$
- Compare x to $a_{n/2}$ - if equal return yes
- Call binary search on list of size $n/2$

(base case: $n = 0$ or 1)

Runtime: Let $B(k)$ = runtime of binary search on a list of size k

$B(0) = O(1)$, $B(1) = O(1)$

$B(n) = \underset{O(1)}{3} + B\left(\frac{n}{2}\right)$

Merge Sort

O(n)

- Divide list into 2 smaller lists of size $n/2$
- Recursively sort smaller lists
- "Merge" list together:
  Each comparison between 2 smallest is the 2 lists moves one item to the sorted list

$\leq n$ comparisons

↳ ~~$\leq 5n$~~ operations $O(n)$

Runtime: Let $M(k)$ = runtime on list of size k

$$M(n) = M\left(\frac{n}{2}\right) + M\left(\frac{n}{2}\right) + O(n)$$
$$= 2M\left(\frac{n}{2}\right) + O(n)$$

$M(1) = O(1)$, $M(0) = O(1)$

Solving these:

Clearly, our former methods to solve won't work, since these are not linear recurrences.

We'll see a powerful technique called recursion trees, although we'll also summarize in the result in Master thm.

But first...

$$B(k) = B\left(\frac{k}{2}\right) + 1$$

Unrolling:

Ex: $B(n) = B\left(\frac{n}{2}\right) + 1$, $B(1) = 1$

$= B\left(\frac{n}{4}\right) + 1 + 1$ ($k = \frac{n}{2}$)

$= B\left(\frac{n}{8}\right) + 1 + 1 + 1$

$= B\left(\frac{n}{2^i}\right) + \underbrace{1 + 1 + 1 \cdots + 1}_{i}$

know $\frac{n}{2^d} = 1$

$n = 2^d$

$\Rightarrow \log_2 n = d$

$= B\left(\frac{n}{2^d}\right) + \underbrace{1 + \cdots + 1}_{d} = \log_2 n + 1$

$$T(k) = 3T\left(\frac{k}{2}\right) + (k-2)$$

Try this on more complex one:

$$T(n) = 3T\left(\frac{n}{2}\right) + (n-2)$$

$$= 3\left(3T\left(\frac{n}{4}\right) + \frac{n}{2} - 2\right) + (n-2)$$

$$= 3\left(3\left(3T\left(\frac{n}{8}\right) + \frac{n}{4} - 2\right) + \frac{n}{2} - 2\right) + n - 2$$

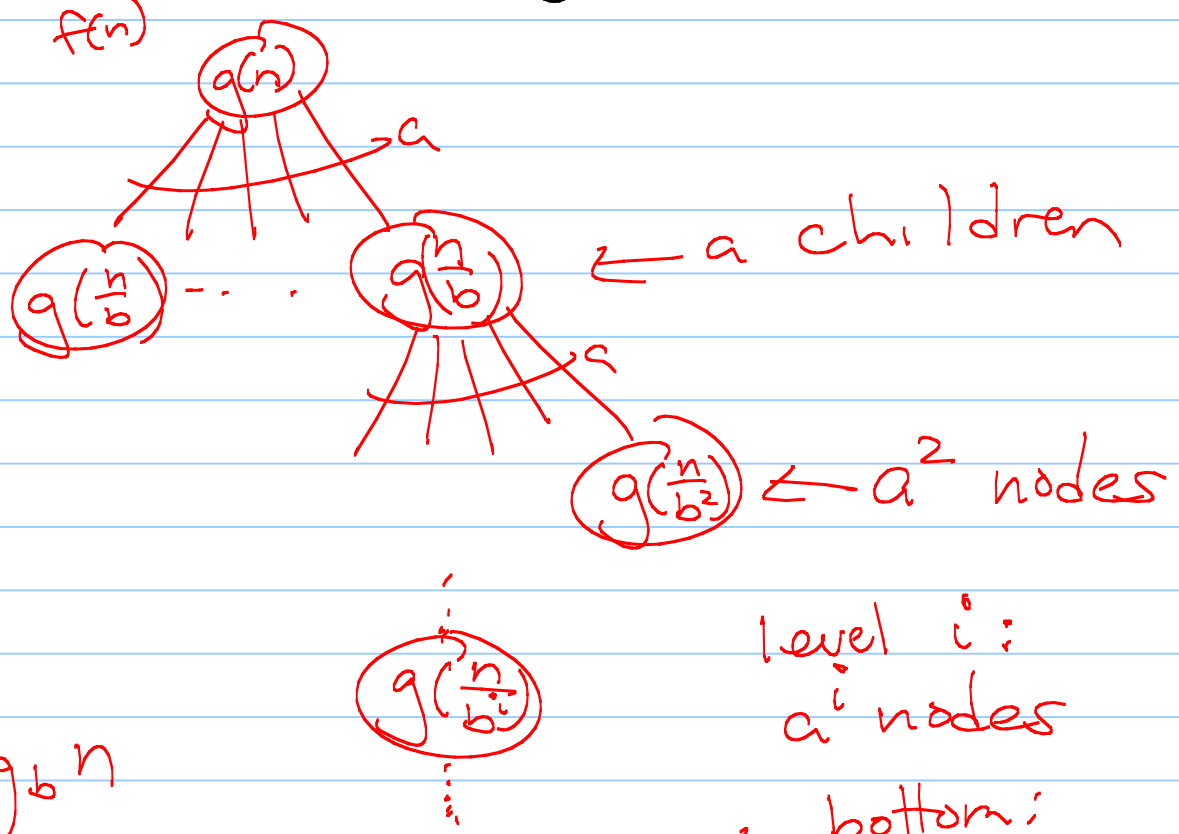
Let's take a step back: Unroll.

$$f(n) = af\left(\frac{n}{b}\right) + g(n)$$

$f(k) = af(\frac{k}{b}) + g(k)$

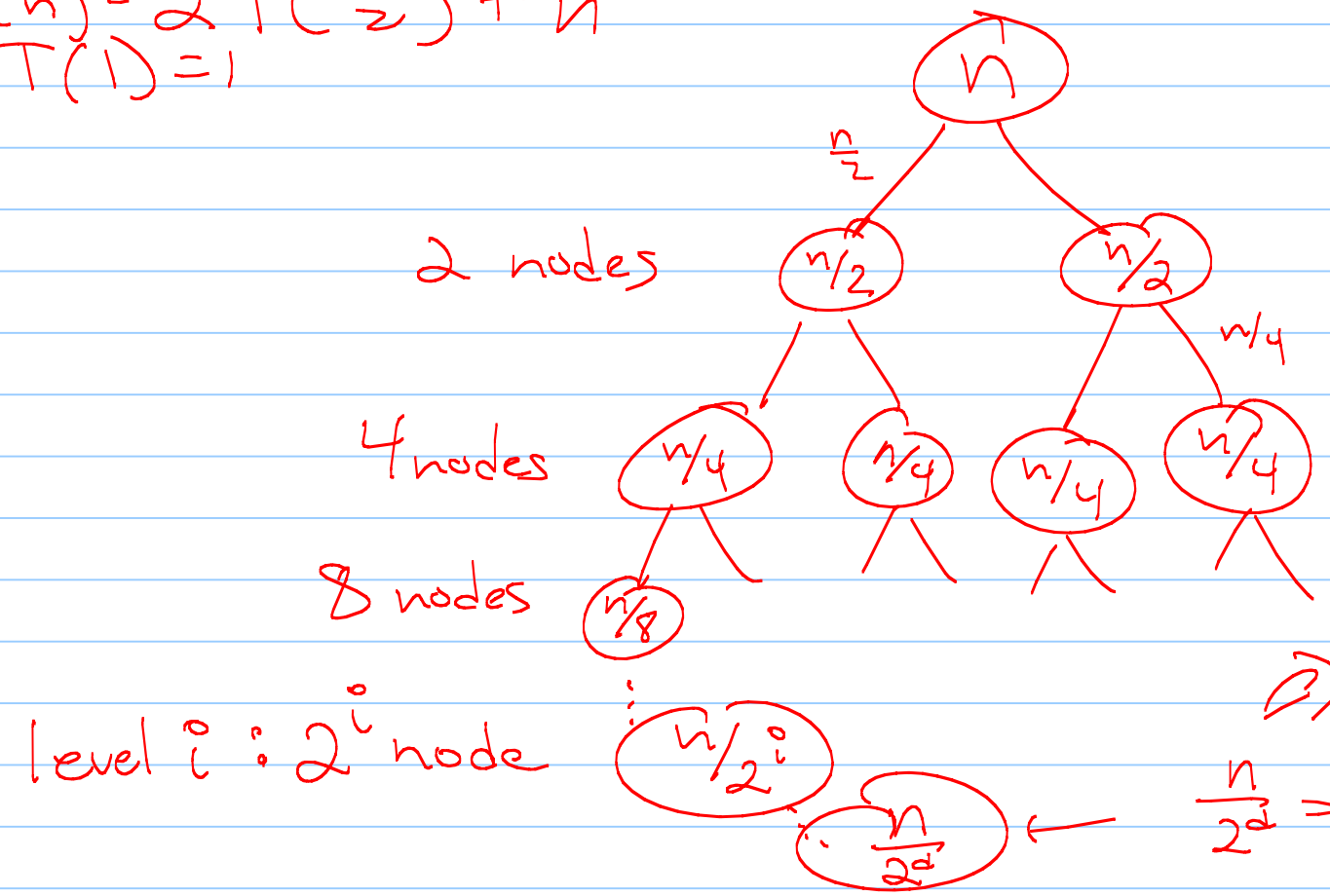
Better way: $f(n) = af(\frac{n}{b}) + g(n)$, $f(1) = 1$

Build a tree:



$\frac{n}{b^d} = 1 \Rightarrow d = \log_b n$

$$T(n) = 2T\left(\frac{n}{2}\right) + n$$
$$T(1) = 1$$



Then to get total "work", sum over all levels in the tree:

$$T(n) = \sum_{i=0}^{d} (\#\text{ nodes at level } i)(\text{work in each node at level } i)$$

$$= \sum_{i=0}^{\log_2 n} (2^i)\left(\frac{n}{2^i}\right)$$

$$= \sum_{i=0}^{\log_2 n} n = n\left(\sum_{i=0}^{\log_2 n} 1\right) = n(\log_2 n + 1)$$

$$= O(n \log_2 n)$$

for $f(n) = af\left(\frac{n}{b}\right) + g(n)$:

$$\sum_{i=0}^{\text{depth}} \left(\begin{array}{c}\#\text{ nodes} \\ \text{on level } i\end{array}\right)\left(\begin{array}{c}\text{work in} \\ \text{each node} \\ \text{at level } i\end{array}\right)$$

$$= \sum_{i=0}^{\log_b n} \left(a^i\right)\left(g\left(\frac{n}{b^i}\right)\right)$$

In some cases, ↑ is a geometric series.