CSE 202 - Algorithms

Recurrences

Master Method

Your turn ...

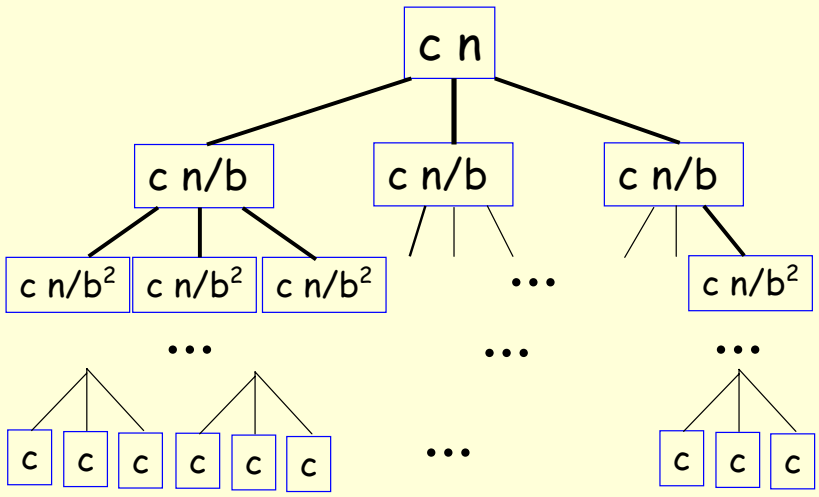
For <u>one</u> of the following recursion trees ...

1. $T(n) = 3T(n/4) + 5n$ for $n = 256$
2. $T(n) = 2T(n/2) + 5n$ for $n = 32$
3. $T(n) = 3T(n/2) + 5n$ for $n = 32$
4. $T(n) = 3T(n/2) + n^2$ for $n = 32$
5. $T(n) = 4T(n/2) + n^2$ for $n = 32$

...figure out the total "combine time" (the last term) needed at each level.

Answer should be a sequence of 5 or 6 numbers.

Recursion Tree for $T(n) = aT(n/b) + cn$



1 depth-0 node

a depth-1 nodes

a^2 depth-2 nodes

...

$a^{\log_b n}$ nodes at depth $\log_b n$

$$T(n) < c\,(\,1 + a(n/b) + a^2(n/b^2) + \ldots + a^{\log_b n}(n/\,b^{\log_b n})\,)$$

$$< cn\,(\,1 + a/b + (a/b)^2 + \ldots + (a/b)^{\log_b n}\,).$$

How does your tree grow?

What's $T(n) = cn\,(\,1 + a/b + (a/b)^2 + \ldots + (a/b)^{\log_b n}\,)$?

The largest term of a geometric series "dominates".

If $a/b < 1$, the first term dominates

Thus, $T(n) \in \Theta(n)$.

If $a/b > 1$, the last term dominates

So $T(n) \in \Theta(n(a/b)^{\log_b n}) = \Theta(n(a^{\log_b n}/b^{\log_b n})) = \Theta(n(a^{\log_b n}/n))$
$= \Theta(a^{\log_b n}) = \Theta(n^{\log_b a})$.

If $a/b = 1$, all terms are equal. There are $\log_b n$ terms.
So $T(n) \in \Theta(n \log_b n) = \Theta(n \lg n)$.

Where are we ??

In Divide & Conquer ... if $T(n) = aT(n/b) + cn$,

(i.e. if you can combine pieces with *linear* work)

then there are three cases:

if $a>b$, then $T(n)$ is $\Theta(n^{\log_b a})$ (there are so many tiny subproblems, they dominate the time)

if $a=b$, then $T(n)$ is $\Theta(n \lg n)$ (just like merge sort)

if $a<b$, then $T(n)$ is $\Theta(n)$ (big step is most expensive)

What if combining takes f(n) work??

In Divide & Conquer ... if $T(n) = aT(n/b) + f(n)$, then three corresponding cases are:

1. The tiny subproblems dominate the run time
   - Happens when $f(n) < c\, a\, f(n/b)$ for some $c<1$ and all n
   - If so, $T(n) \in \Theta(a^{\log_b n}) = \Theta(n^{\log_b a})$.
2. All levels take about the same time
   - Happens when $f(n)$ is $\Theta(a^{\log_b n})$.
   - If so, $T(n) = \Theta(f(n) \lg n)$.
3. Big step is most expensive
   - Happens when $f(n) > c\, a\, f(n/b)$ for some $c>1$ and all n.
   - If so, $T(n) = \Theta(f(n))$.

Previous slide is "Master Method"

Slight differences:

Book's condition on case 1, "f(n) is $O(n^{\log_b a-\epsilon})$", is slightly more general.

It allows f(n) to be less uniform.

Case 2 – remember $a^{\log_b n} = n^{\log_b a}$.

Book has (unnecessary) extra condition in case 3

"f(n) is $\Omega(n^{\log_b a+\epsilon})$" is implied by "f(n) > c a f(n/b), c>1"

Master method can "interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$"

Other recurrences

Master Method doesn't always apply:

What if, in MergeSort, we divided 75% - 25% ?

$T(n) = T(3n/4) + T(n/4) + c\,n.$

Or we divided into 1000 and n-1000 sized pieces?

$T(n) = T(1000) + T(n-1000) + cn$ (for n>1000).

Or consider:

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