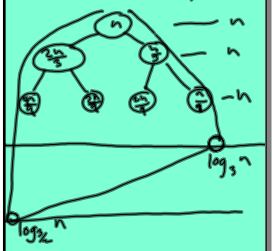
Tree method (more examples)
Master method

$T(n) = 4T(\frac{n}{2}) + n^2$

$+$

$T(n) = \Theta(n^2 \lg n)$

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



$T(n) \geq n \log_3 n$

$T(n) \leq n \log_2 n$

$\log_b n = \Theta(\log_c n)$

$\forall b, c > 1.$

$\therefore T(n) = \Theta(n \lg n)$

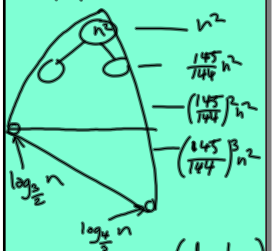
$T(n) = T(\frac{2n}{3}) + T(\frac{3n}{4}) + n^2$

$T(n) = \Omega(n^{2+\varepsilon})$

for some $\varepsilon > 0$.

$= n^2(\frac{4}{9} + \frac{9}{16})$

$= \frac{145}{144} n^2$



lower bound: $(d := \log_{\frac{3}{2}} n)$

$T(n) \geq \sum_{i=0}^{d} (\frac{145}{144})^i n^2$

$= n^2 \left(\frac{(\frac{145}{144})^{d+1} - 1}{\frac{145}{144} - 1} \right)$

$= 144 n^2 \left((\frac{145}{144})^{d+1} - 1 \right)$

$= \Omega\left(n^2 (\frac{145}{144})^d \right)$

$= \Omega\left(n^2 (\frac{145}{144})^{\log_{\frac{3}{2}} n} \right)$

$= \Omega\left(n^2 \, n^{\log_{\frac{3}{2}} (\frac{145}{144})} \right)$

$= \Omega\left(n^{2 + \underbrace{\log_{\frac{3}{2}} (\frac{145}{144})}_{\varepsilon > 0}} \right)$

Sep 13-12:49 PM

Upper bound:

$T(n) = O\left(n^{2 + \log_3(\frac{1+n}{1+n})}\right)$

Not a tight bound.

Master method

Solves recurrences of the form

$T(n) = aT\left(\frac{n}{b}\right) + f(n)$

for $a \geq 1$, $b > 1$, and $f(n)$ fairly well-behaved.

Theorem (Master Theorem)
Let $a \geq 1$ and $b > 1$ be real constants. Let $f(n)$ be eventually positive, and let $T(n)$ be given by the recurrence
$T(n) = aT\left(\frac{n}{b}\right) + f(n)$.

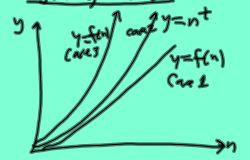
Fix $t := \log_b a$.

Then:
Case 1: If $f(n) = O(n^s)$ for some constant $s < t$, then $T(n) = \Theta(n^t)$.

Case 2: If $f(n) = \Theta(n^t)$, then $T(n) = \Theta(n^t \lg n)$

Case 3: If $f(n) = \Omega(n^u)$ for some constant $u > t$, and $af(n/b) \leq c \cdot f(n)$ for some constant $c < 1$ and for all large enough n, then $T(n) = \Theta(f(n))$.

regularity condition



Merge sort recurrence
$T(n) = 2T(\frac{n}{2}) + n$
$a = 2$
$b = 2$
$t = \log_b a = \log_2 2 = 1$
$f(n) = n = n^1 = n^t$
case 2 applies
$T(n) = \Theta(n \lg n)$

$T(n) = 4T(\frac{n}{2}) + n$
(running time of slow mult)
$a = 4$
$b = 2$
$t = \lg 4 = 2$ ($1 < t$)
$f(n) = n = n^1$
case 1: $T(n) = \Theta(n^2)$

Fast mult:
$T(n) = 3T(\frac{n}{2}) + n$

Sep 13-1:10 PM

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

$a = 3$

$b = 2$

$t = \lg 3 \doteq 1.6\ldots$

$f(n) = n^1$ $\quad (1 < 1.6\ldots)$

case 1

$T(n) = \Theta(n^t) = \Theta(n^{\lg 3})$

Fastest known algo for integer mult is $\Theta(n \lg n)$

(uses Fast Fourier Transform FFT)

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

$a = 1$

$b = 2$

$t = \lg 1 = 0$ $\quad (1 > 0)$

$f(n) = n = n^1$

case 3

$T(n) = \Theta(f(n))$

$= \Theta(n)$.

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

$a = 2$

$b = 2$

$t = \lg 2 = 1$

$f(n) = n \lg n = \omega(n)$

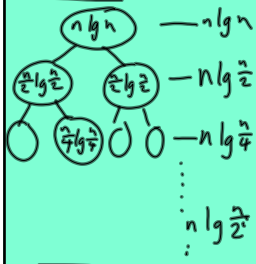
but $f(n) = o(n^{1+\varepsilon})$

for any $\varepsilon > 0$.

between case 2 & case 3

Can't apply Master Thm

Tree Method



$d := \lg n$

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

$= n \sum_{i=0}^{d} \lg \frac{n}{2^i}$

$= n \sum_{i=0}^{d} (\lg n - \lg 2^i)$

$= n\left(\sum_{i=0}^{d} \lg n - \sum_{i=0}^{d} i\right)$

$\doteq n\left(d \lg n - \frac{d^2}{2}\right)$

$= n\left((\lg n)^2 - \frac{(\lg n)^2}{2}\right)$

$= \frac{1}{2} n (\lg n)^2$

$= \Theta\left(n (\lg n)^2\right)$

Do yourself

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

Sep 13-1:30 PM