Recurrences

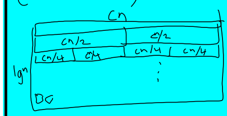
A recurrence is an equation or inequality relating a function $T(n)$ in terms of $T(m)$ for $m < n$.

Ex: $T(n)$ = time to mergesort a list of size n.

$T(0) = T(1) = c$

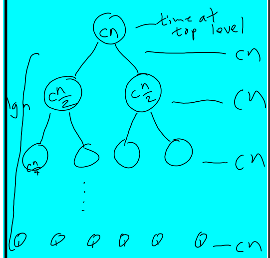
$T(n) = 2T(\frac{n}{2}) + cn$

(constant c)



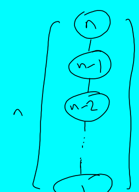
3 techiques to solve recurrences

1. Substitution method — gives a rigorous proof of an asymptotic bound (using strong induction) [does not tell you what the bound is "from scratch"]
2. Tree method: "unwind" the recursion, placing the task times as nodes of a tree. Ex: merge sort:



Ex: Insertion sort

$T(n) = T(n-1) + n$

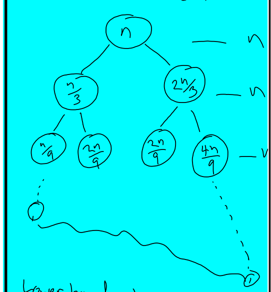


$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$

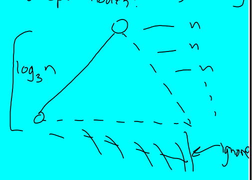
$= \Theta(n^2$

Ex:

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



Lower bound: chop the tree at the shallowest leaf, removing all deeper nodes:



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

Upperbound: take the

Depth $= \log_{\frac{3}{2}} n$

$\therefore T(n) \leq n \log_{3/2} n$

Summary:

$n\log_3 n \leq T(n) \leq n\log_{3/2} n$

but $\log_3 n = \Theta(\log_{\frac{3}{2}} n)$
and vice versa
both are $\Theta(\lg n)$
so multiplying by n:

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

3rd method: Master Theorem

Subst method:
$T(n) = 2T(\frac{n}{2}) + n$

Upper bound (UB):
W.T.S. $T(n) = O(n \lg n)$
Assume (inductive hypothesis)
that $T(m) \leq c\, m \lg m$
for all $m < n$.

some explicit constant c to be determined

$T(n) = 2T(\frac{n}{2}) + n$
$\leq 2\left(c\left(\frac{n}{2}\right)\lg\left(\frac{n}{2}\right)\right) + n$
(ind. hyp.) $\frac{n}{2} < n$

$= cn\lg(\frac{n}{2}) + n$
$= cn(\lg n - \underbrace{\lg 2}) + n$
$= cn(\lg n - 1) + n$
$= cn\lg n - (c-1)n$
$\underset{\text{need}}{\leq} \underbrace{cn\lg n}$. ✓ UB

True provided $c > 1$

Lower bound (LB)
Assume $T(m) \geq c m\lg m$ $(m<n)$
[some constant $c' > 0$ to be determined] (not necessary that $c' = c$)

WTS: $T(n) \geq cn\lg n$
$T(n) = 2T(\frac{n}{2}) + n$
$\geq 2c'(\frac{n}{2})\lg(\frac{n}{2}) + n$
$= \cdots = c'n\lg n - (c'-1)n$
$\geq c'n\lg n$
provided $(c'-1)n \leq 0$
$c' - 1 \leq 0$
$0 < c' \leq 1$ ✓ LB

Issues: $\frac{n}{2}$ is not an integer if n is odd,
Actual recurrence:
$T(n) = T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + n$

[note: $n = \lfloor \frac{n}{2} \rfloor + \lceil \frac{n}{2} \rceil$]

Extreme example;
$T(n) = 2T(\frac{n}{2} + 10) + n$

$T(n) = O(n \lg n)$ in this case.

Assume $T(m) \leq c m\lg m$ $(m<n)$

$$T(n) = 2T\left(\frac{n}{2}+10\right) + n$$

$$\leq 2c\left(\frac{n}{2}+10\right)\lg\left(\frac{n}{2}+10\right)+n$$

(ind. hyp.)

doesn't work if $\frac{n}{2}+10 \geq n$ (!)
(can't apply ind. hyp.)

but $\frac{n}{2}+10 < n$ if $n>20$

Can start the induction is sufficiently

(continuing) (ind. hyp.)

~~large to make things work,~~
~~assuming~~

$$\cancel{\leq 2c\left(\frac{n}{2}+10\right)\lg\left(\frac{n}{2}+10\right)+n}$$

$\because (n>20 \because)$

$$\leq 2c\left(\frac{n}{2}+10\right)\lg\left(\frac{n}{2}+10\right)+n$$

$$= c(n+20)\lg\left(\frac{n}{2}+10\right)+n$$

$$= cn\lg\left(\frac{n}{2}+10\right) + 20c\lg\left(\frac{n}{2}+10\right)+n$$

$$\leq cn\lg\left(\frac{n}{2}+10\right) + 20c\lg n + n$$

$$\leq cn\lg\left(\frac{2n}{3}\right) + 20\lg n + n$$

$$\left(\text{if } \frac{n}{2}+10 \leq \frac{2n}{3} \quad 10 \leq \frac{n}{6} \quad 60 \leq n\right)$$

$$= cn\lg n + cn\underbrace{\lg\tfrac{2}{3}}_{<0} + 20\lg n + n$$

$\leq cn\lg n$ provided

$$cn\lg\frac{2}{3} + 20\lg n + n \leq 0$$

$$cn(1-\lg 3) + 20\lg n + n \leq 0$$

$$cn\underbrace{(1-\lg 3)}_{<0} \leq -20\lg n - n$$

$$c \geq \frac{-20\lg n - n}{n(1-\lg 3)} = \frac{n+20\lg n}{n(\lg 3 - 1)} = \frac{1}{\lg 3 - 1} + \frac{20\lg n}{n(\lg 3 - 1)}$$

pick n big enough so that

$$\frac{1}{\lg 3 - 1} + \frac{20\lg n}{n(\lg 3 - 1)} \leq \frac{1}{\lg 3 - 1} + 1$$

set $c := \frac{1}{\lg 3 - 1} + 1$