Math 182 Lecture 8

§4.2 The substitution method cont.

Recall: Substitution method

1. Guess form of sol.
2. Use mathematical induction to find constants and show sol. works.

Last time: $T(n) = 2T(\lfloor n/2 \rfloor) + n$

showed $T(n) = O(n \lg n)$

substitution method gave us condition $c \geq 1$.

A counterintuitive trick

Example: $T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1$

satisfies $T(n) = O(n)$

Want to show there is

Discussion ~~Sps there is~~ $c > 0$ s.t. $T(n) \leq cn$, eventually. Sps there $c > 0$ s.t. $T(m) \leq cm$ for $m < n$.

$$\begin{aligned} T(n) &= T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1 \\ &\leq c\lfloor n/2 \rfloor + c\lceil n/2 \rceil + 1 \\ &= cn + 1 \end{aligned}$$

want $\rightarrow \leq cn$. No value of c works :(

Trick: change inductive assumption by subtracting a lower order term.

Want to show there are $c, d > 0$ s.t.

$$T(n) \le cn - d \quad \forall n$$

Sps we have such c, d and $T(m) \le cm - d$ for $m < n$.

$$T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1$$

$$\le c\lfloor n/2 \rfloor - d + c\lceil n/2 \rceil - d + 1$$

$$= cn - 2d + 1$$

$$\le cn - d$$

want $\rightarrow$

need $-2d + 1 \le -d \rightarrow 1 \le 2d \rightsquigarrow d \ge 1/2$

so any $c > 0$, $d \ge 1/2$ will make inductive step work. ✓

Changing Variables Consider

$$T(n) = 2T(\lfloor \sqrt{n} \rfloor) + \lg n.$$

Will show $T(n) = O(\lg n \lg\lg n)$

Discussion: Consider instead

$$T(n) = 2T(\sqrt{n}) + \lg n$$

Change vars: $\lg n =: m$

$\rightsquigarrow$ $T(2^m) = 2T(2^{m/2}) + m$

Define $S(m) := T(2^m)$

$\rightsquigarrow S(m) = 2S(m/2) + m$

Know: $S(m) = O(m \lg m)$

Thus $T(n) = T(2^n) = S(m) = O(m \lg m)$

$= O(\lg n \lg \lg n)$. ✓

General asymptotic trick:

consider substituting

$\lg n$ for n

or 2^n for n

---

§ The recursion-tree method

This is useful for:

- Getting a good guess at asymptotic upper / lower bound
- Seeing the global picture of total work done by recursive algorithm.

Example $T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2)$

satisfies $T(n) = \Theta(n^2)$.

Discussion: Note: $T(n) = \Omega(n^2)$ bcs

have to show / find upper bound.

Simplify to

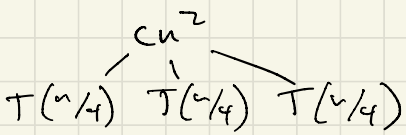
$T(n) = 3T(n/4) + cn^2$ for some $c > 0$.

(assume n is power of 4)

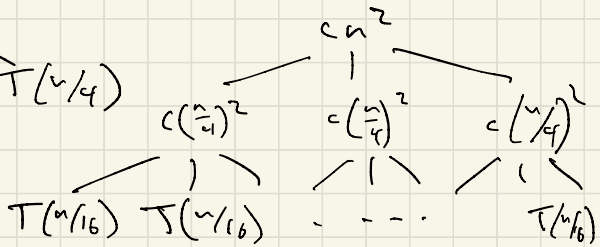
No levels

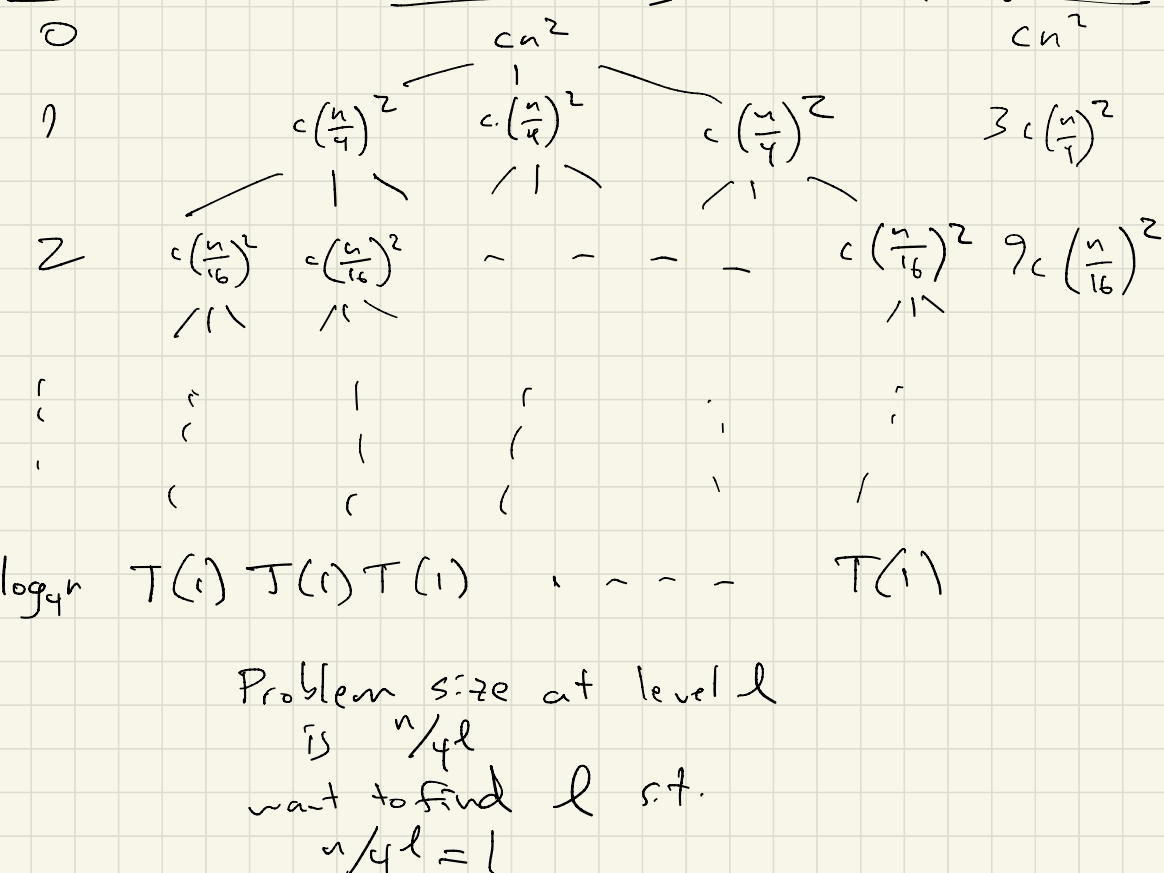
$T(n)$

1 level



2 levels





Problem size at level l is $n/4^l$

want to find l s.t. $n/4^l = 1$

$\rightsquigarrow n = 4^l \rightsquigarrow l = \log_4 n$

nodes per level $= 3^l$

cost per node on level $l = c(\frac{n}{4^l})^2$

cost per level for $l \le \log_4 n - 1 = 3^l c(\frac{n}{4^l})^2$

$= cn^2(\frac{3}{16})^l$

Cost for bottom row

$= 3^{\log_4 n} \cdot T(1) = T(1) n^{\log_4 3} = \Theta(n^{\log_4 3})$

Total cost:

$$T(n) = \sum_{\ell=0}^{\log_4 n - 1} cn^2 \left(\frac{3}{16}\right)^\ell + \Theta(n^{\log_4 3})$$

$$= \frac{(3/16)^{\log_4 n} - 1}{(3/16) - 1} cn^2 + \Theta(n^{\log_4 3})$$

$$\underbrace{\qquad\qquad}_{\gamma}$$

Since only need upper bound, use infinite sum:

$$T(n) = \sum_{\ell=0}^{\log_4 n - 1} cn^2 \left(\frac{3}{16}\right)^\ell + \Theta(n^{\log_4 3})$$

$$< cn^2 \sum_{\ell=0}^{\infty} \left(\frac{3}{16}\right)^\ell + \Theta(n^{\log_4 3})$$

$$= cn^2 \frac{1}{1 - 3/16} + \Theta(n^{\log_4 3})$$

$$= c\frac{16}{13} n^2 + \Theta(n^{\log_4 3})$$

$$= O(n^2) \quad \checkmark$$

↑ guess

Example The recurrence

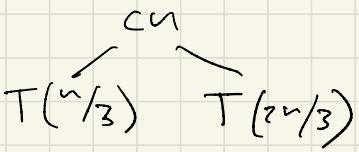
$$T(n) = T(n/3) + T(2n/3) + O(n)$$

satisfies $O(n \lg n)$.

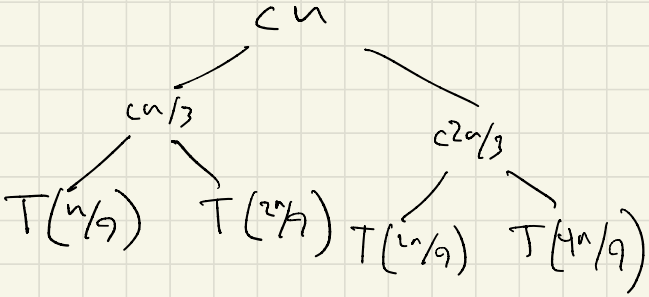
Discussion: Can simplify:

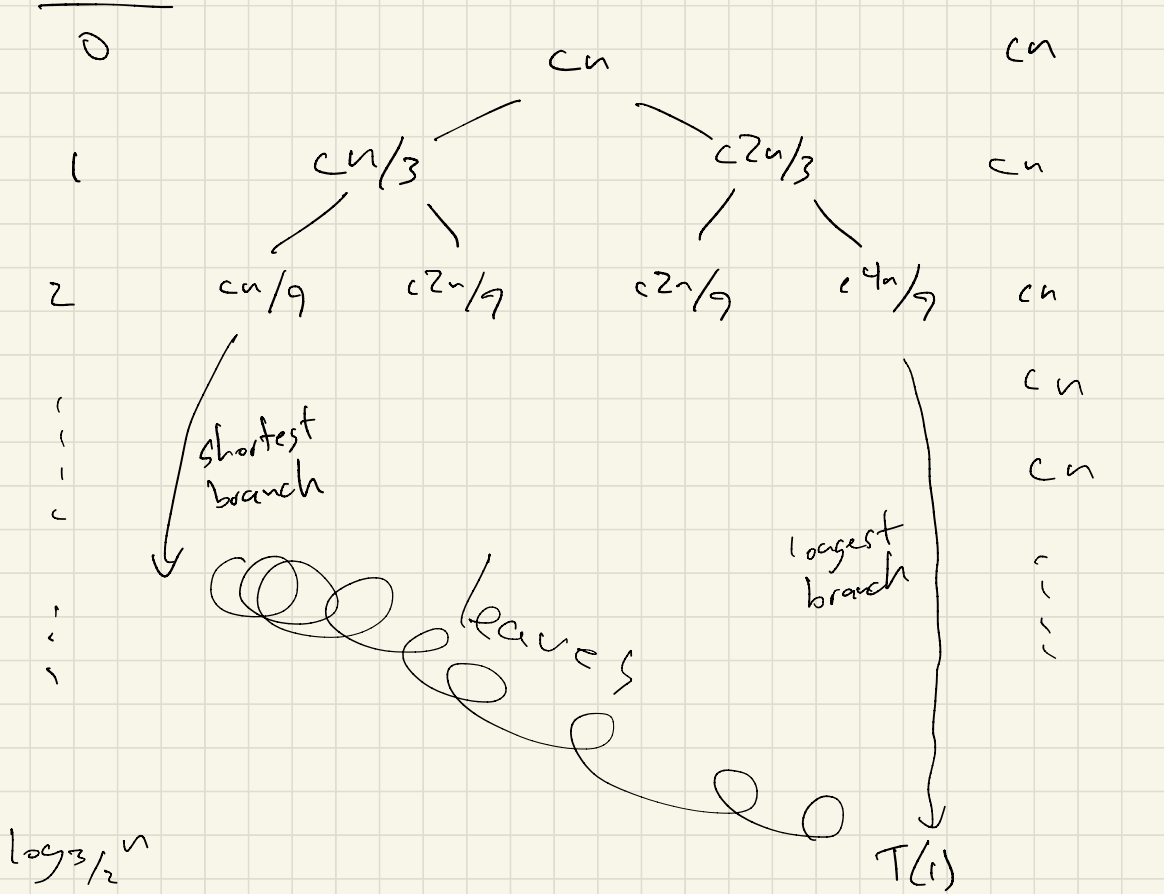
$$T(n) = T(n/3) + T(2n/3) + cn \quad (c > 0)$$

One level



Two levels





On level ℓ subproblem size on right branch is $= n\left(\frac{2}{3}\right)^{\ell}$

want $n(2/3)^{\ell} = 1$

$\rightsquigarrow \ell = \log_{3/2} n$

Guess: (of upper bound): Assume simplest thing w/o considering leaves: every row has cost cn and $\log_{3/2} n + 1$ levels

Guess: $T(n) = O(cn \log_{3/2} n + cn) = O(n \lg n)$.

To check if guess works: need to show $\exists d > 0$ s.t. event. $T(n) \leq dn\lg n$.

Assume we have such a d and $T(m) \leq dm\lg m$ for $m < n$.

$$T(n) \leq T(n/3) + T(2n/3) + cn$$
$$\leq d(n/3)\lg(n/3) + d(2n/3)\lg 2n/3 + cn$$
$$\rightarrow = dn\lg n - dn(\lg 3 - 2/3) + cn$$

want $\rightarrow \leq dn\lg n$

need: $-dn(\lg 3 - 2/3) + cn \leq 0$

$\rightsquigarrow$ $d \geq c/(\lg 3 - 2/3)$

s. for any c always can find d that works. ✓

Thus $T(n) = O(n\lg n)$.

§4.4 The Master Method

$3\,T(n/4)$

$$T(n) = a\,T(n/b) + f(n)$$

$a \geq 1$ — # subproblems

$b > 1$ — n/b subproblem size

$f(n)$ — amount done at root

$\left(n^{\log_b a}\right) \lessgtr f(n)$

Master Theorem 4.4.1. *Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be an asymptotically positive function, and let $T(n)$ be defined on the nonnegative integers by the recurrence*

$$T(n) = aT(n/b) + f(n),$$

where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ has the following asymptotic bounds:

(1) (Leaf-heavy) If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.

(2) (Middle case) If $f(n) = \Theta(n^{\log_b a} \lg^k n)$ for some constant k, then

- *(a) if $k > -1$, then $T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$,*
- *(b) if $k = -1$, then $T(n) = \Theta(n^{\log_b a} \lg \lg n)$, and*
- *(c) if $k < -1$, then $T(n) = \Theta(n^{\log_b a})$.*

(3) (Root-heavy) If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant ϵ, and the following holds:

- *(Regularity Condition) if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n,*

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