Solving Recurrences

Lecture 1 → Merge Sort → Analysis produced recurrence

$$T(n) = 2T(n/2) + cn \qquad (\text{for } c > 0)$$

These are so common that will develop approaches to figuring out what this means

→ $\Theta(n \lg n)$ ("theta")

We will do this commonly

6.001 → recursive solutions were common

★ L3 → divide and conquer approaches

~~How to solve recurrences~~

Goals: Given recurrence expression — describes asymptotic behavior for large n

① Find order of Growth

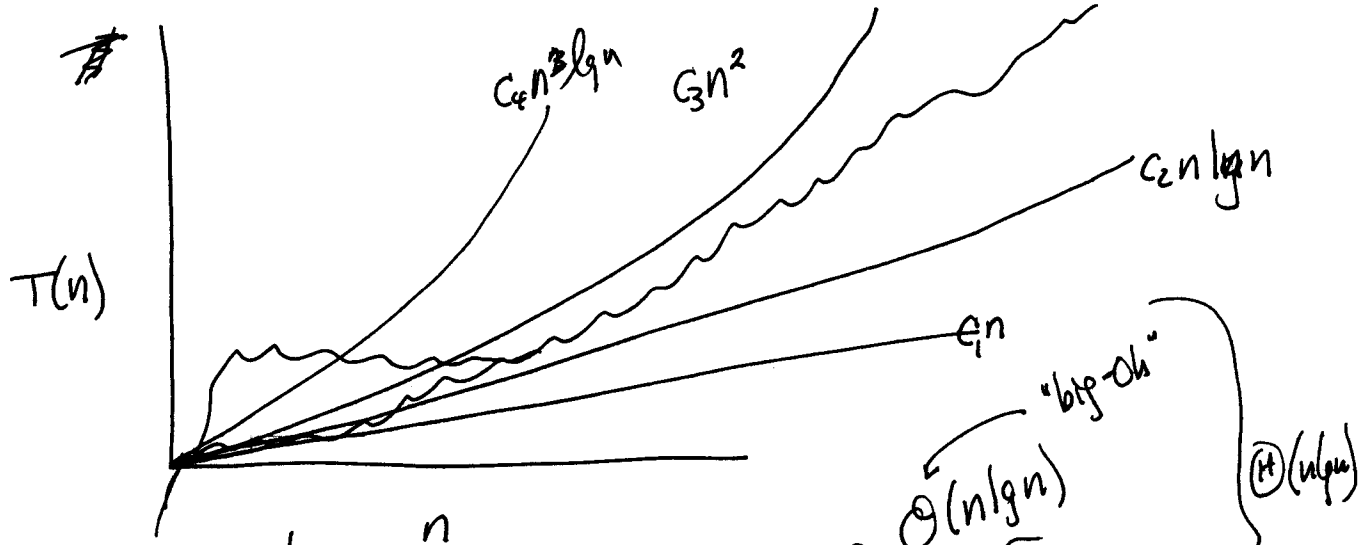
② Prove order of Growth

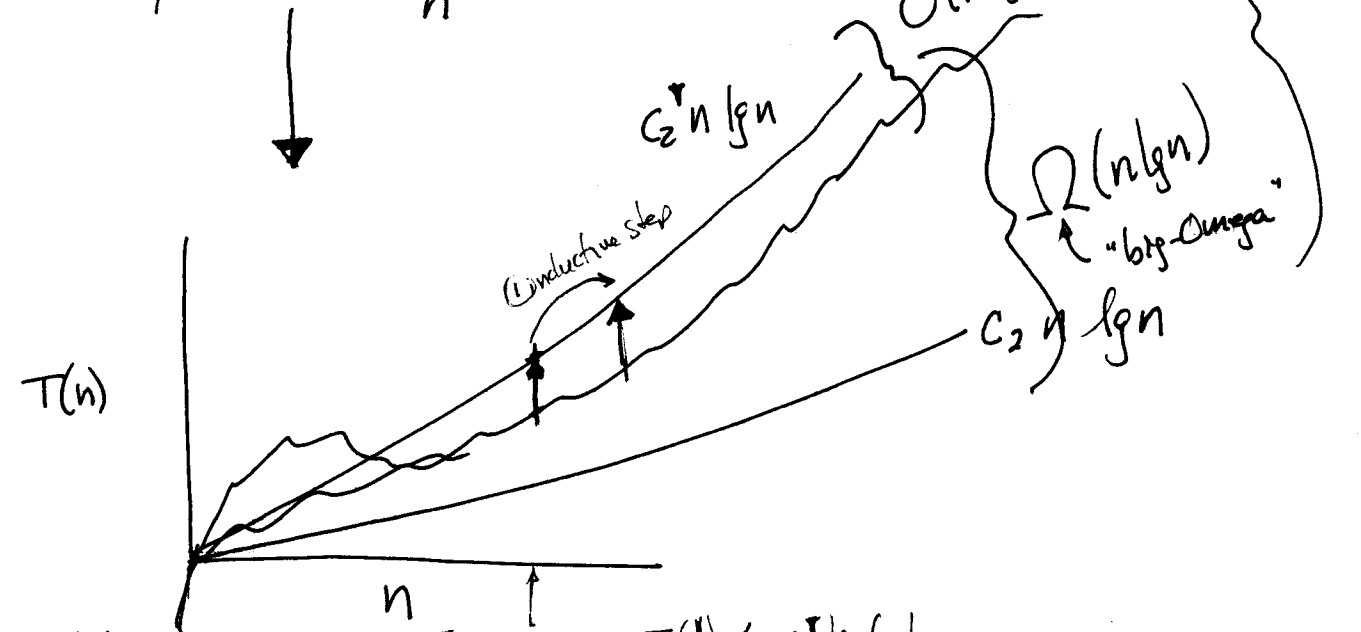
Approaches: → combination of intuitive understanding and learning some tricks (like solving diff eq)

I Substitution Method — Mathematical Induction → prove

II Recurrence Recursion-Tree Method → find → prove by substitution

III Master Method → find & prove
- memorize rules
- proven by theorem, for certain forms

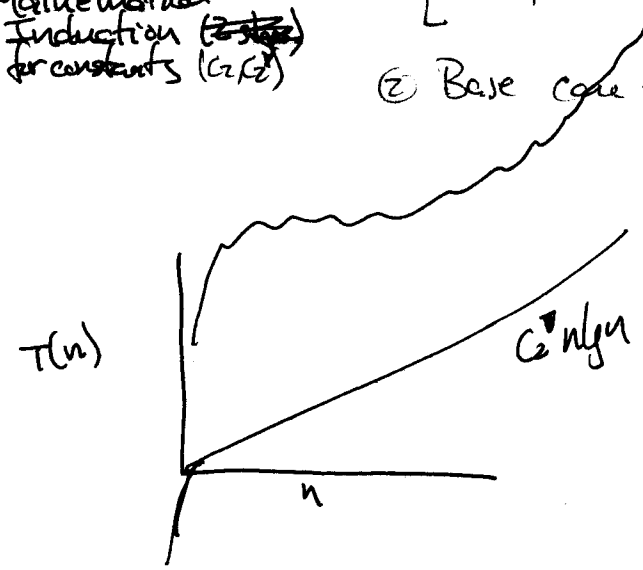




(I) ~~The Substitution Method~~
- "Guess" ~~form of~~ solution
- Prove Correct by Mathematical Induction ~~(2 steps)~~
- Solve for constants (c_2, c_2')

(1) (a) assume $T(k) \le c_2' k \lg k$ } prove
(b) prove $T(n) \le c_2' n \lg n$ for $n > k$

(2) Base case: demonstrate $T(n) \le c_2' n \lg n$ for some small value of n



Example: Merge-Sort $T(n) = 2T(n/2) + n$ $\xrightarrow{\text{Guess}}$ $T(n) = \Theta(n \lg n)$

~~Need to~~

Solution:
- Need to prove O & Ω separately — we'll just do O here
- Will need $T(n)$ for small n; let's assume $T(1) = \Theta(1) = 1$

Inductive Step:

① Assume ~~$T(n) = 2T(n/2) + n$~~ $T(k) \le ck \lg k$ for $k < n$ and $c > 0$ ← This is our guess

② Substitute into recurrence

$T(n) = 2T(n/2) + n$ (let $k = n/2$)

$\le 2c\frac{n}{2}\lg\frac{n}{2} + n$

$= cn \lg n - (cn \lg 2 - n)$ (cn)

$= \underbrace{cn \lg n}_{\text{desired}} - \underbrace{(c-1)n}_{\text{residual}}$

$\le cn \lg n$ for $c \ge 1$ ✓

Base Case

③ ~~Solve for~~ Demonstrate Base Case ("Boundary Cond")

$T(1) = \Theta(1) = 1$ ← substitute into recurrence

~~$T(1) = c \cdot 1 \cdot \lg 1 = 0$ but want $\Theta(1) \le T(1)$~~

$T(1) = \Theta(1) = 1$

$\le c \cdot 1 \cdot \lg 1$

$= 0$ ✗ Doesn't work

Problem is that our proof need only be for large n, but can't demonstrate for $n = 1$. Solution is to increase base case.

$T(2) = 2T(n/2) + n = 2T(1) + 2 = 4$ ← assume $T(1) = 1$ ✓

~~$T(2) = 4$~~ $\le c \cdot 2 \lg 2 = c2$ → true for $c \ge 2$ ✓

Guessing Strategies

① Recurrence similar to one that's solved
→ Guess ~~similar~~ some soln

ex: $T(n) = 2T(\frac{n}{2}+17) + n$

↑ only difference from above (will be insignificant for large n)

Guess $O(n) = n \lg n$ & verify w/ substitution meth.
also helps w/ floors and ceilings

② Work out lower & upper bounds separately and converge

- lower bound $\Omega(n)$
- upper bound $O(n^2)$

} → guess $\Theta(n \lg n)$ although n & n^2 are still possible

Subtleties

① Correct guess, but can't prove induction.
→ Soln: often need to reuse guess by subtracting a lower-order term.

Ex: $T(n) = 4T(n/2) + n$ → Guess $O(n^2)$

Assume $T(k) \le ck^2$ for $k < n$ and $c > 0$

$T(n) = 4T(n/2) + n$ ↓ $k = n/2$

$\le 4c(\frac{n}{2})^2 + n$

$= cn^2 + n = \underbrace{cn^2}_{\text{desired}} - \underbrace{(-n)}_{\text{residual}}$

$\le cn^2$ only for $-n > 0$ or $n < 0$ ← doesn't work!!

for any value of $c > 0$

don't stop here and claim $O(n^2)$. MUST PROVE INDUCTIVE HYPOTHESIS

soln: Revise guess to $T(k) \le c_1 k^2 - c_2 k$ for $k < n$ & $c_1, c_2 > 0$

$$T(n) = 4T(n/2) + n$$
$$\le 4\left(c_1 \left(\frac{n}{2}\right)^2 - c_2\left(\frac{n}{2}\right)\right) + n$$
$$= c_1 n^2 - 2c_2 n + n = (c_1 n^2 - c_2 n) - (c_2 n - n)$$
$$\le c_1 n^2 - c_2 n \quad \text{for } c_2 > 1$$ ✓ Fixes Problem

Select c_1 large enough to handle base case

Lesson, if induction fails, don't just reuse bound; for instance asymptotic behavior $O(n^3)$ or $n^2 \lg n$

Why does this work? By decreasing the function, it looks like It seems that we should increase our guess rather than decrease. Because the guess is also the assumption in the inductive step, it turns out that a smaller guess gives a tighter bound and a stronger inductive hypothesis.

② Another useful trick: Changing Variables

ex: $T(n) = 2T(\lfloor \sqrt{n} \rfloor) + \lg n$ — another subtlety

$T(2^m) = 2T(2^{m/2}) + m$ — let $m = \lg n \to n = 2^m$

$S(m) = 2S(m/2) + m$ — let $S(m) = T(2^m)$

$S(m) = \Theta(m \lg m)$ substitution changes back $T(n) = \Theta(\lg n \lg(\lg n))$

(II) Recursion-Tree Method

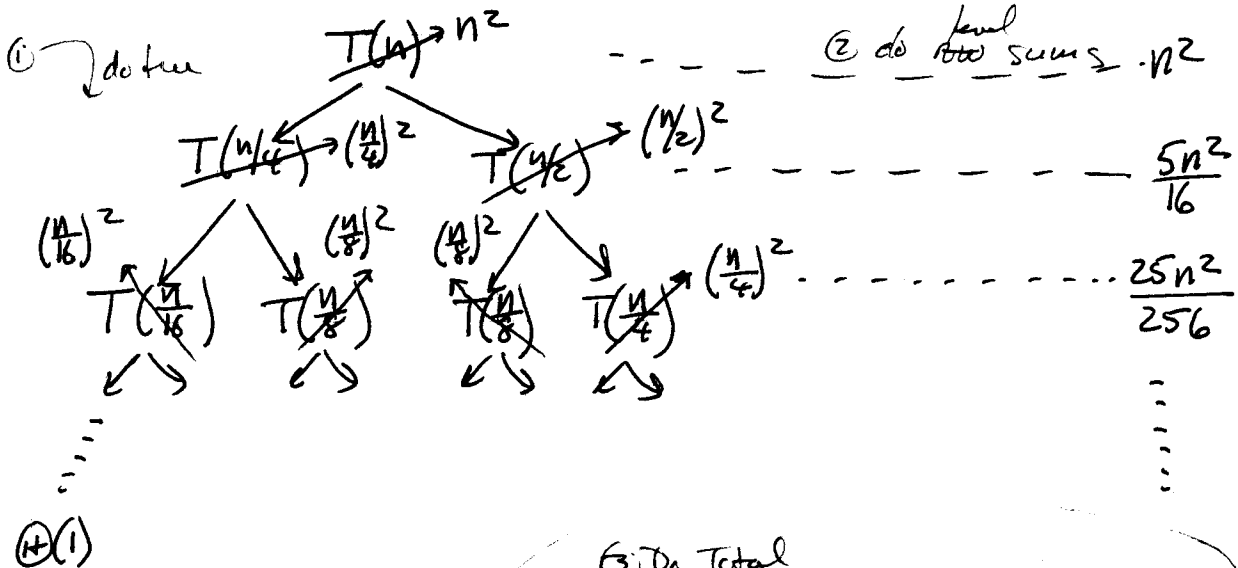
Approach

① Build Recursion Tree — each node is cost of single subproblem
② Sum cost of each level
③ Sum all levels

→ Best as guess for substitution method (but if done carefully enough, could serve as proof)

→ Great for generating intuition

Ex: $T(n) = T(n/4) + T(n/2) + n^2$



③ Do Total

$$\text{Total} = n^2\left(1 + \frac{5}{16} + \left(\frac{5}{16}\right)^2 + \left(\frac{5}{16}\right)^3 + \cdots + \left(\frac{5}{16}\right)^k\right)$$

$\Rightarrow \Theta(n^2)$

# of terms related to height of tree

not quite since tree has ragged bottom.

Don't erase.

$1 + x + x^2 + \cdots x^k = \frac{1-x^{k+1}}{1-x}$ for $x \neq 1$

$1 + x + x^2 + \cdots = \frac{1}{1-x}$ for $|x| < 1$

This term doesn't grow with k, because even if tree were infinite $\cdots 1 + x + x^2 + \cdots = \frac{1}{1-x} \Rightarrow \frac{16}{11}$

← This tree is dominated by its root node and not by its depth.

(III) The Master Method (formulaic)

Applies to recurrences of form: $T(n) = a\,T(n/b) + f(n)$ where $a \geq 1$, $b > 1$, and f is asymptotically positive.

Three Cases

① If $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$
THEN $T(n) = \Theta(n^{\log_b a})$

height of tree → # of leaves

② If $f(n) = \Theta(n^{\log_b a})$
THEN $T(n) = \Theta(n^{\log_b a} \cdot \lg n)$

product →

③ If $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some constant $\varepsilon > 0$,
and if $a\,f(n/b) \leq c\,f(n)$ for some constant $c < 1$ and all sufficiently large n
THEN $T(n) = \Theta(f(n))$

root node →

regularity condition →

④ None of the above ⇒ Master method does not apply.

~~What's this all about?~~

- Intuitively, all 3 cases compare $f(n)$ to ~~$n^{\log_b a}$~~
  ~~Case 1: if $n^{\log_b a}$ is larger~~

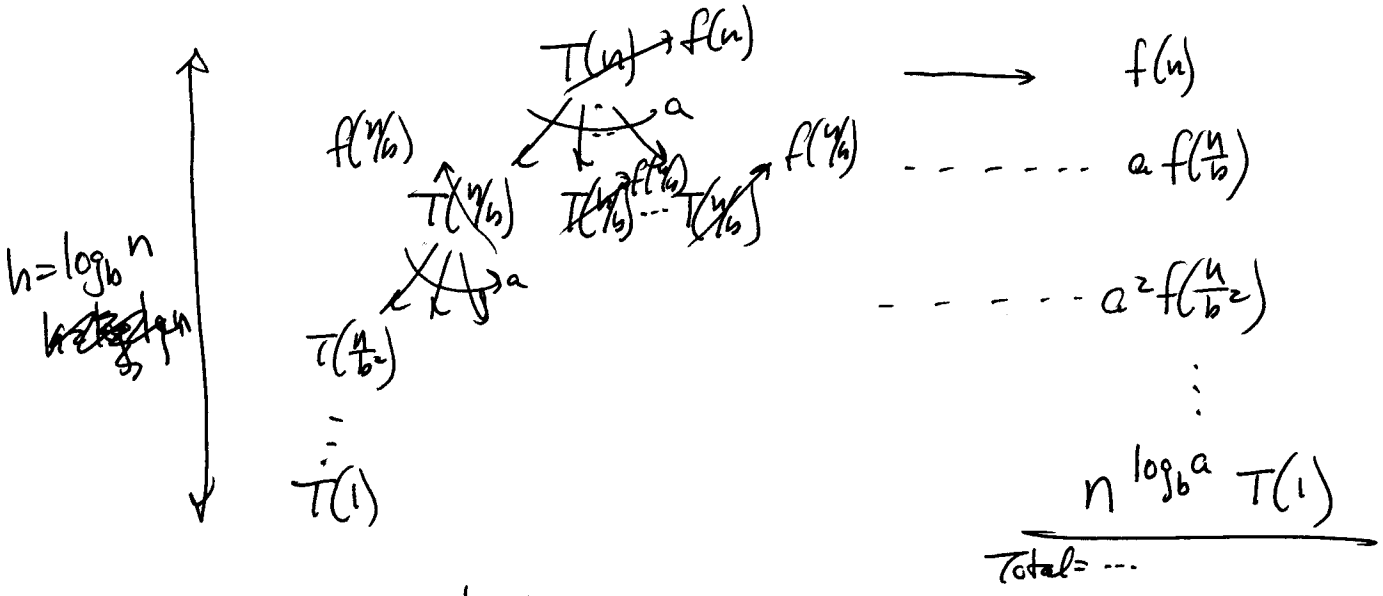
~~$f(n)$~~

Proof, not required, ☞ §4.4 of book

What's going on here ...

All 3 cases compare $f(n)$ to $n^{\log_b a}$

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



① Case 1: $n^{\log_b a}$ is larger than $f(n)$ [in a polynomial sense, with ε]
that the ~~bottom row of~~ leaves (bottom row) dominates total
[total grows as march down levels]

③ Case 2: $n^{\log_b a}$ similar enough to $f(n)$, that both contribute, so multiply weight of leaves by height of tree
[total about same at each level]

② Case 3: $f(n)$ is larger than $n^{\log_b a}$, so root node dominates total
[total for each level decreases down tree]

Examples:

Ex 1: $T(n) = 4T(n/2) + n$

$a = 4 \quad b = 2 \quad n^{\log_b a} = n^2; \quad f(n) = n$

$f(n) = O(n^{2-\varepsilon})$ for $\varepsilon = 1 \Rightarrow$ Case 1

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

Ex 2: $T(n) = 4T(n/2) + n^2$

$a = 4 \quad b = 2 \quad n^{\log_b a} = n^2 \quad f(n) = n^2$

$f(n) = \Theta(n^2) \Rightarrow$ Case 2

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

Ex 3: $T(n) = 4T(n/2) + n^3$

$a = 4 \quad b = 2 \quad n^{\log_b a} = n^2 \quad f(n) = n^3$

$f(n) = \Omega(n^{2+\varepsilon})$ for $\varepsilon = 1$

and $4\left(\frac{n}{2}\right)^3 \le cn^3$ for $c = 1/2$ $\Rightarrow$ Case 3

$\therefore T(n) = \Theta(n^3)$

Ex 4: $T(n) = 4T(n/2) + \frac{n^2}{\lg n}$

$a = 4 \quad b = 2 \quad n^{\log_b a} = n^2 \quad f(n) = \frac{n^2}{\lg n}$

Master method does not apply because

$n^{\varepsilon} = \omega(\lg n)$ for $\varepsilon > 0$

~~wait~~ ~~$\frac{n^3}{\lg n} = O(n^{2-\varepsilon})$~~

~~$n^{\varepsilon} \lg n \frac{1}{\lg n} = O(n^{-\varepsilon})$~~

~~$\lg n \neq 0$~~

$n^{\varepsilon} = O(\lg n)$

Falls between cases 1 & 2

More General Method (Akra-Bazzi)

$$T(n) = \sum_{i=1}^{k} a_i T(n/b_i) + f(n)$$

Let p be unique soln to $\sum_{i=1}^{k} \left(\frac{a_i}{b_i^p} \right) = 1$

Then cases are same as Master Method with n^p replacing $n^{\log_b a}$.

(Akra & Bazzi also prove even more general results).

- See chapter notes in book & end of chap 4.