Design & Analysis of Algorithms
COMP 482 / ELEC 420

John Greiner

Math Background: Review & Beyond

1. Asymptotic notation
2. Math used in asymptotics
3. Recurrences
4. Probabilistic analysis

To do:
[CLRS] 4
#2

Obtaining Recurrences

Introductory multiplication examples:

T(n) = O(1)	n=1	T'(n) = O(1)	n=1
T(n) = 4T(n/2) + kn	n>1	T'(n) = 3T'(n/2) + k'n	n>1

Obtained from straightforward reading of algorithms.

Key observation: Deterministic algorithms lead to recurrences.

1. Determine appropriate metric for the size "n".
2. Examine how metric changes during recursion/iteration.

Solving for Closed Forms

T(n) = O(1)	n=1	T'(n) = O(1)	n=1
T(n) = 4T(n/2) + kn	n>1	T'(n) = 3T'(n/2) + k'n	n>1

$T(n) = \Theta(n^2)$ $\quad$ $T'(n) = \Theta(n^{\log_2 3})$

How?
In general, hard. Solutions not always known.
Will discuss techniques in a few minutes…

Recurrences' Base Case

T(n) = O(1)	**n<b**
T(n) = …	n≥b

For constant-sized problems, can bound algorithm by some constant.

This constant is irrelevant for asymptote. Often skip writing base case.

Recurrences

T(n) = 4T(n/2) + kn	n>1	T'(n) = 3T'(n/2) + k'n	n>1

? What if n is odd? ¿

Next iteration, n is not integral. Nonsense.

T(n) = 3T(⌈n/2⌉) + T(⌊n/2⌋) + kn	n>1

Above more accurate.
The difference rarely matters, so usually ignore this detail.

Two Common Forms of Recurrences

Divide-and-conquer: $T(n) = a \times T(n/b) + f(n) \quad n \geq b$

Linear: $T(n) = a_1 T(n-1) + a_2 T(n-2) + f(n) \quad n > b$

Techniques for Solving Recurrences

- Substitution
- Recursion Tree
- Master Method – for divide & conquer
- Summation – for simple linear
- Characteristic Equation – for linear

Techniques: Substitution

Guess a solution & check it.

More detail:

1. Guess the form of the solution, using unknown constants.
2. Use induction to find the constants & verify the solution.

Completely dependent on making reasonable guesses.

Substitution Example 1

$T(n) = 4T(n/2) + n \quad n>1$

Simplified version of previous example.

Guess: $T(n) = O(n^3)$.

More specifically:
$T(n) \leq cn^3$, for all large enough n.

Substitution Example 1

$T(n) = 4T(n/2) + n \quad n>1$

Guess:
$T(n) \leq cn^3$ for $\forall n>n_0$

Prove by strong induction on n.

? Which means what exactly? ?

Assume: $T(k) \leq ck^3$ for $\forall k>n_0$, for $\forall k<n$.
Show: $T(n) \leq cn^3$ for $\forall n>n_0$.

Substitution Example 1

$T(n) = 4T(n/2) + n \quad n>1$

Guess:
$T(n) \leq cn^3$ for $\forall n>n_0$

Assume $T(k) \leq ck^3$ for $\forall k>n_0$, for $\forall k<n$. Show $T(n) \leq cn^3$ for $\forall n>n_0$.

Base case, $n=n_0+1$?

Awkward. Fortunately, $n_0=0$ works in these examples.

Substitution Example 1

$T(n) = 4T(n/2) + n \quad n>1$

Guess:
$T(n) \le cn^3$

Assume $T(k) \le ck^3$, for $\forall k<n$. Show $T(n) \le cn^3$.

Base case, n=1:

$T(n) = 1$	Definition.
$1 \le c$	Choose large enough c for conclusion.

Substitution Example 1

$T(n) = 4T(n/2) + n \quad n>1$

Guess:
$T(n) \le cn^3$

Assume $T(k) \le ck^3$, for $\forall k<n$. Show $T(n) \le cn^3$.

Inductive case, n>1:

$T(n) = 4T(n/2) + n$	Definition.
$\le 4c(n/2)^3 + n$	Induction.
$= c/2 \times n^3 + n$	Algebra.

While this is $O(n^3)$, we're not done.
Need to show $c/2 \times n^3 + n \le c \times n^3$.
Fortunately, the constant factor is shrinking, not growing.

Substitution Example 1

$T(n) = 4T(n/2) + n \quad n>1$

Guess:
$T(n) \le cn^3$

Assume $T(k) \le ck^3$, for $\forall k<n$. Show $T(n) \le cn^3$.

Inductive case, n>1:

$T(n) \le c/2 \times n^3 + n$	From before.
$= cn^3 - (c/2 \times n^3 - n)$	Algebra.
$\le cn^3$	For n>0, if $c \ge 2$.

Substitution Example 1

$T(n) = 4T(n/2) + n \quad n>1$

Proved: $T(n) \le 2n^3$ for $\forall n>0$

Thus, $T(n) = O(n^3)$.

Substitution Example 2

$T(n) = 4T(n/2) + n \quad n>1$

Guess: $T(n) = O(n^2)$.
Same recurrence, but now try tighter bound.

More specifically:
$T(n) \le cn^2$ for $\forall n>n_0$.

Substitution Example 2

$T(n) = 4T(n/2) + n \quad n>1$

Guess:
$T(n) \le cn^2$ for $\forall n>n_0$

Follow same steps, and we get...

Assume $T(k) \le ck^2$, for $\forall k<n$. Show $T(n) \le cn^2$.

$T(n) = 4T(n/2) + n$
$\le 4c(n/2)^2 + n$
$= cn^2 + n$

Not $\le cn^2$!
Problem is that the constant isn't shrinking.

Substitution Example 2

$T(n) = 4T(n/2) + n \qquad n>1$

Solution: Use a tighter guess & inductive hypothesis.

Subtract a lower-order term – a common technique.

Guess:
$T(n) \le cn^2 - dn$ for $\forall n>0$

Assume $T(k) \le ck^2 - dk$, for $\forall k<n$. Show $T(n) \le cn^2 - dn$.

Substitution Example 2

$T(n) = 4T(n/2) + n \qquad n>1$

Guess:
$T(n) \le cn^2 - dn$

Assume $T(k) \le ck^2 - dn$, for $\forall k<n$. Show $T(n) \le cn^2 - dn$.

Base case, n=1:

$T(n) = 1$	Definition.
$1 \le c-d$	Choosing c, d appropriately.

Substitution Example 2

$T(n) = 4T(n/2) + n \qquad n>1$

Guess:
$T(n) \le cn^2 - dn$

Assume $T(k) \le ck^2 - dn$, for $\forall k<n$. Show $T(n) \le cn^2 - dn$.

Inductive case, n>1:

$T(n) = 4T(n/2) + n$	Definition.
$\le 4(c(n/2)^2 - d(n/2)) + n$	Induction.
$= cn^2 - 2dn + n$	Algebra.
$= cn^2 - dn - (dn - n)$	Algebra.
$\le cn^2 - dn$	Choosing $d \ge 1$.

Substitution Example 2

$T(n) = 4T(n/2) + n \qquad n>1$

Proved: $T(n) \le 2n^2 - 1n$ for $\forall n>0$

Thus, $T(n) = O(n^2)$.

Techniques: Recursion Tree

Guessing correct answer can be difficult!
Need a way to obtain appropriate guess.

1. Unroll recurrence to obtain a summation.
2. Solve or estimate summation.
3. Use solution as a guess in substitution.

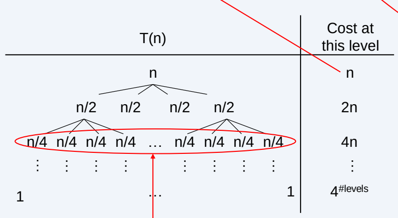
Math sometimes tricky.

Recursion Tree Example 1

$T(n) = 4T(n/2) + n \qquad n>1$

? How many levels? ?

$\log_2 n$

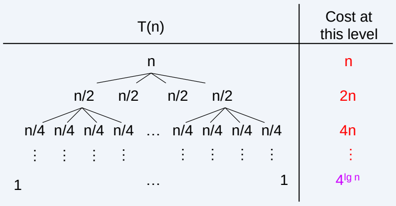


Now, turn picture into a summation…

In this example, all terms on a level are the same.
Common, but not always true.

Recursion Tree Example 1

$T(n) = 4T(n/2) + n \qquad n>1$



$$T(n) = n\left(\sum_{i=0}^{\lg n-1} 2^i\right) + n^2$$
$$= n\left(\frac{2^{\lg n}-1}{2-1}\right) + n^2$$
$$= n\left(\frac{2^{\lg n}}{2}\right) + n^2$$
$$= n\left(\frac{n^{\lg 2}}{2}\right) + n^2$$
$$= n\left(\frac{n}{2}\right) + n^2$$
$$= \Theta(n^2)$$

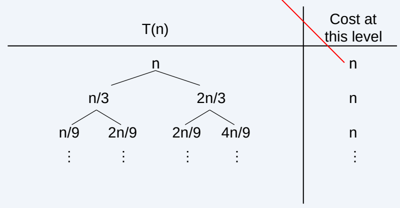
$T(n) = n + 2n + 4n + \ldots + 2^{\lg n - 1}n + 4^{\lg n}$
$= n(1 + 2 + 4 + \ldots + 2^{\lg n - 1}) + n^{\lg 4}$

Recursion Tree Example 2

$T(n) = T(n/3) + T(2n/3) + n \qquad n>1$

How many levels?

$\log_{3/2} n$



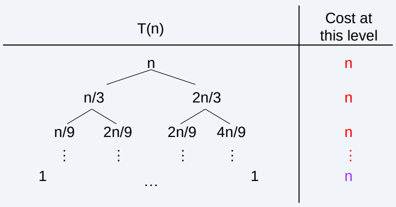
But, not all branches have same depth!

Makes cost near the leaves hard to calculate.

Estimate!

Recursion Tree Example 2

$T(n) = T(n/3) + T(2n/3) + n \qquad n>1$



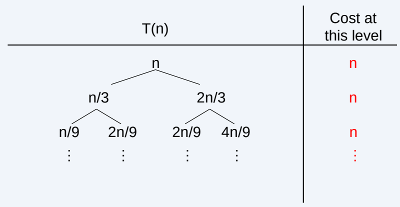
Overestimate.
Consider all branches to be of max depth.

$T(n) \leq n(\log_{3/2} n - 1) + n$
$T(n) = O(n \log n)$

#levels $= \log_{3/2} n$

Recursion Tree Example 2

$T(n) = T(n/3) + T(2n/3) + n \qquad n>1$



Underestimate.
Count the complete levels, & ignore the rest.

$T(n) \geq n(\log_3 n - 1)$
$T(n) = \Omega(n \log n)$

#levels $= \log_3 n$

Thus, $T(n) = \Theta(n \log n)$

Techniques: Master Method

Cookbook solution for <u>many</u> recurrences of the form

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

where

$a \geq 1$, $b>1$, $f(n)$ asymptotically positive

First describe its cases, then outline proof.

Master Method Case 1

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

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

$T(n) = 7T(n/2) + cn^2 \qquad a=7, b=2$

E.g., Strassen matrix multiplication.

$cn^2 \stackrel{?}{=} O(n^{\log_b a - \varepsilon}) = O(n^{\log_2 7 - \varepsilon}) \approx O(n^{2.8 - \varepsilon})$

Yes, for any $\varepsilon \leq 0.8$.

$T(n) = \Theta(n^{\lg 7})$

Master Method Case 2

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

$f(n) = \Theta(n^{\log_b a}) \rightarrow T(n) = \Theta(n^{\log_b a} \lg n)$

$T(n) = 2T(n/2) + cn$ a=2, b=2

E.g., mergesort.

$cn =^? \Theta(n^{\log_b a}) = \Theta(n^{\log_2 2}) = \Theta(n)$

Yes.

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

Master Method Case 3

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

$f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some $\varepsilon>0$ and

$a \times f(n/b) \leq c \times f(n)$ for some $c<1$ and all large enough n

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

I.e., is the constant factor shrinking?

$T(n) = 4T(n/2) + n^3$ a=4, b=2

$n^3 =^? \Omega(n^{\log_b a + \varepsilon}) = \Omega(n^{\log_2 4 + \varepsilon}) = \Omega(n^{2 + \varepsilon})$

Yes, for any $\varepsilon \leq 1$.

$4(n/2)^3 = ½ \times n^3 \leq^? cn^3$

Yes, for any $c \geq ½$. $T(n) = \Theta(n^3)$

Master Method Case 4

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

None of previous apply. Master method doesn't help.

$T(n) = 4T(n/2) + n^2/\lg n$ a=4, b=2

Case 1?

$n^2/\lg n =^? O(n^{\log_b a - \varepsilon}) = O(n^{\log_2 4 - \varepsilon}) = O(n^{2 - \varepsilon}) = O(n^2/n^\varepsilon)$

No, since lg n is asymptotically < n^ε.

Thus, $n^2/\lg n$ is asymptotically > n^2/n^ε.

Master Method Case 4

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

None of previous apply. Master method doesn't help.

$T(n) = 4T(n/2) + n^2/\lg n$ a=4, b=2

Case 2?

$n^2/\lg n =^? \Theta(n^{\log_b a}) = \Theta(n^{\log_2 4}) = \Theta(n^2)$

No.

Master Method Case 4

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

None of previous apply. Master method doesn't help.

$T(n) = 4T(n/2) + n^2/\lg n$ a=4, b=2

Case 3?

$n^2/\lg n =^? \Omega(n^{\log_b a + \varepsilon}) = \Omega(n^{\log_2 4 + \varepsilon}) = \Omega(n^{2 + \varepsilon})$

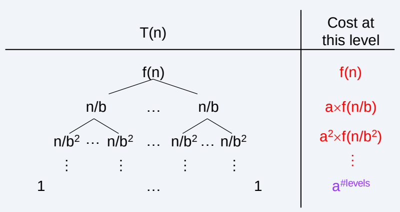
No, since 1/lg n is asymptotically < n^ε.

Master Method Proof Outline

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

? How many levels? ?

$\log_b n$



$$T(n) = \sum_{i=0}^{\log_b n - 1} a^i f(n/b^i) + n^{\log_b a}$$

Cases correspond to determining which term dominates & how to compute sum.

Technique: Summation

For linear recurrences with one recursive term.

$T(n) = T(n-1) + f(n)$ ⇨ ?

$T(n) = T(n-2) + f(n)$ ⇨ ?

Techniques: Characteristic Equation

Applies to linear recurrences

- Homogenous:
  $a_n = c_1a_{n-1} + c_2a_{n-2} + \ldots + c_ka_{n-k}$
- Nonhomogenous:
  $a_n = c_1a_{n-1} + c_2a_{n-2} + \ldots + c_ka_{n-k} + F(n)$