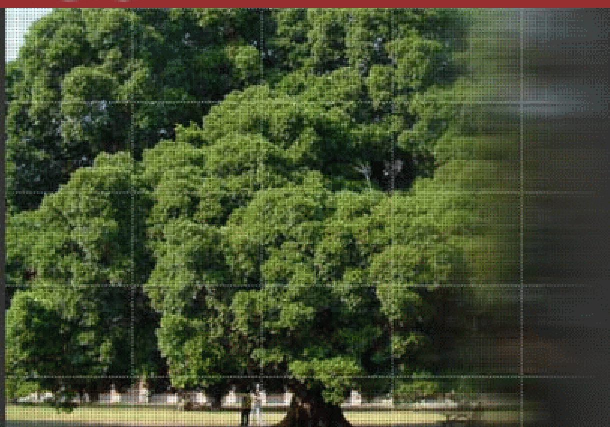
National Cheng Kung University



Chapter 4: Recurrences

Sun-Yuan Hsieh

謝孫源 教授

成功大學資訊工程學系

Overview

A recurrence is a function is defined in terms of

- one or more base cases, and
- itself, with smaller arguments

Examples:

$$T(n)=\begin{cases}1 & \text{if } n=1\\ T(n-1)+1 & \text{if } n>1\end{cases}$$

Solution: $T(n)=n.$

$$T(n)=\begin{cases}1 & \text{if } n=1\\ 2T(n/2)+n & \text{if } n>1\end{cases}$$

Solution: $T(n)=n\lg n+n.$

$$T(n)=\begin{cases}0 & \text{if } n=2\\ T(\sqrt{n})+1 & \text{if } n>2\end{cases}$$

Solution: $T(n)=\lg\lg n.$

$$T(n)=\begin{cases}1 & \text{if } n=1\\ T(n/3)+T(2n/3)+n & \text{if } n>1\end{cases}$$

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

Many technical issues:

- *Floors and ceilings*

  [Floors and ceilings can easily be removed and don’t affect the solution to the recurrence.]
- *Exact vs. asymptotic functions*
- *Boundary condition*

In algorithm analysis, we usually **express both the recurrence and its solution using asymptotic notation**.

- E.g. $T(n) = 2T(n/2) + \Theta(n)$, with solution $T(n) = \Theta(n \lg n)$
- The boundary conditions are usually expressed as "$T(n) = O(1)$ for sufficiently small n."
- When we desire an exact, rather than an asymptotic, solution, we need to deal with boundary conditions.
- In practice, we just use asymptotic most of the time, and we ignore boundary conditions.

Substitution method

1. Guess the solution.
2. Use induction to find the constants and show that the solution works.

E.g.

$$T(n) = \begin{cases} 1 & \text{if } n = 1, \\ 2T(n/2) + n & \text{if } n > 1. \end{cases}$$

1. *Guess:* $T(n) = n\lg n + n$. [Here, we have a recurrence with an exact function, rather than asymptotic notation, and the solution is also exact rather than asymptotic. We'll have to check boundary conditions and the base case.]

Substitution method (cont'd)

2. *Induction*:

   Bass: $n = 1 \Rightarrow n\lg n + n = 1 = T(n)$

   Inductive step: Inductive hypothesis is that $T(k) = k\lg k + k$ for all $k < n$.

   We'll use this inductive hypothesis for $T(n/2)$.

$$
\begin{aligned}
T(n) &= 2T\left(\frac{n}{2}\right) + n \\
&= 2\left(\frac{n}{2}\lg\frac{n}{2} + \frac{n}{2}\right) + n \\
&= n\lg\frac{n}{2} + n + n \\
&= n(\lg n - \lg 2) + n + n \\
&= n\lg n - n + n + n. \\
&= n\lg n + n
\end{aligned}
$$

Substitution method (cont'd)

Generally, we use asymptotic notation:

$$T(n) = 2T(n/2) + \Theta(n)$$

- Assume

$$T(n) = O(1) \text{ for sufficiently small } n$$

- Express the solution by asymptotic notation:

- Don't worry about boundary cases, nor do we show base cases in the substitution proof.

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

Substitution method (cont'd)

- $T(n)$ is always constant for any constant n.
- Since we are ultimately interested in asymptotic solution to a recurrence, it will always be possible to choose base cases that work
- When we want an asymptotic solution to a recurrence, we don't worry about the base cases in our proofs.
- When we want an exact solution, then we have to deal with base cases.

Substitution method (cont’d)

For the substitution method:

- Name the constant in the additive term
- Show the upper (O) and lower (Ω) bounds separately.

Might need to use different constants for each notation

***E.g.*:** $T(n) = 2T(n/2) + \Theta(n)$. If we want to show an upper bound of $T(n)$, we write $T(n) \le 2T(n/2) + cn$ for some positive constant c.

1. **Upper bound**:

Guess: $T(n) \le dn\lg n$ for some positive constant d. We are given c in the recurrence, and we get to choose d as any positive constant. It's OK for d to depend on c.

Substitution:

$$
\begin{aligned}
T(n) &\le 2T(n/2) + cn \\
&= 2\left(d\frac{n}{2}\lg\frac{n}{2}\right) + cn \\
&= dn\lg\frac{n}{2} + cn \\
&= dn\lg n - dn + cn \\
&\le dn\lg n \qquad \text{if } -dn + cn \le 0, \\
&\qquad\qquad\qquad d \ge c
\end{aligned}
$$

Therefore, $T(n) = O(n\lg n)$

2. Lower bound:

Write $T(n) \ge 2T(n/2) + cn$ for some positive constant c.

Guess: $T(n) \ge dn \lg n$ for some positive constant d.

Substitution:

$$
\begin{aligned}
T(n) &\ge 2T(n/2) + cn \\
&\ge 2\left(d\,\frac{n}{2}\lg\frac{n}{2}\right) + cn \\
&= dn\lg\frac{n}{2} + cn \\
&= dn\lg n - dn + cn \\
&\ge dn\lg n \qquad \text{if } -dn + cn \ge 0, \\
& \qquad\qquad\qquad d \le c
\end{aligned}
$$

Therefore, $T(n) = \Omega(n \lg n)$.

Therefore, $T(n) = \Theta(n \lg n)$ [For this particular recurrence, we can use $d = c$ for both the upper-bound and lower-bound proofs. That won't always be the case.]

Substitution method (cont'd)

Make sure you show the same exact form when doing a substitution proof.

Consider the recurrence

$T(n) = 8T(n/2) + \Theta(n^2)$.

For an upper bound:

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

$$T(n) \le 8T(n/2) + cn^2$$

$$\begin{aligned} T(n) &\le 8d(n/2)^3 + cn^2 \\ &= 8d(n^3/8) + cn^2 \\ &= dn^3 + cn^2 \\ &\not\le dn^3 \end{aligned}$$

doesn't work!

Substitution method (cont'd)

Remedy: *Subtract off* a lower-order term.

Guess: $T(n) \le d\, n^3 - d'\, n^2$

$$
\begin{aligned}
T(n) &\le 8(d\,(n/2)^3 - d'(n/2)^2) + cn^2 \\
&= 8d\,(n^3/8) - 8d'\,(n^2/4) + cn^2 \\
&= d\, n^3 - 2\, d'\, n^2 + cn^2 \\
&\le d\, n^3 - d'\, n^2 \quad \text{if } -2d'n^2 + cn^2 \le -d'\, n^2, \\
&\qquad\qquad\qquad\qquad d' \ge c
\end{aligned}
$$

Substitution method (cont'd)

Be careful when using asymptotic notation.

The false proof for the recurrence $T(n) = 4T(n/4) + n$, that $T(n) = O(n)$:

$T(n) \leq 4(c(n/4)) + n$

$\leq cn + n$

$= O(n)$ wrong!

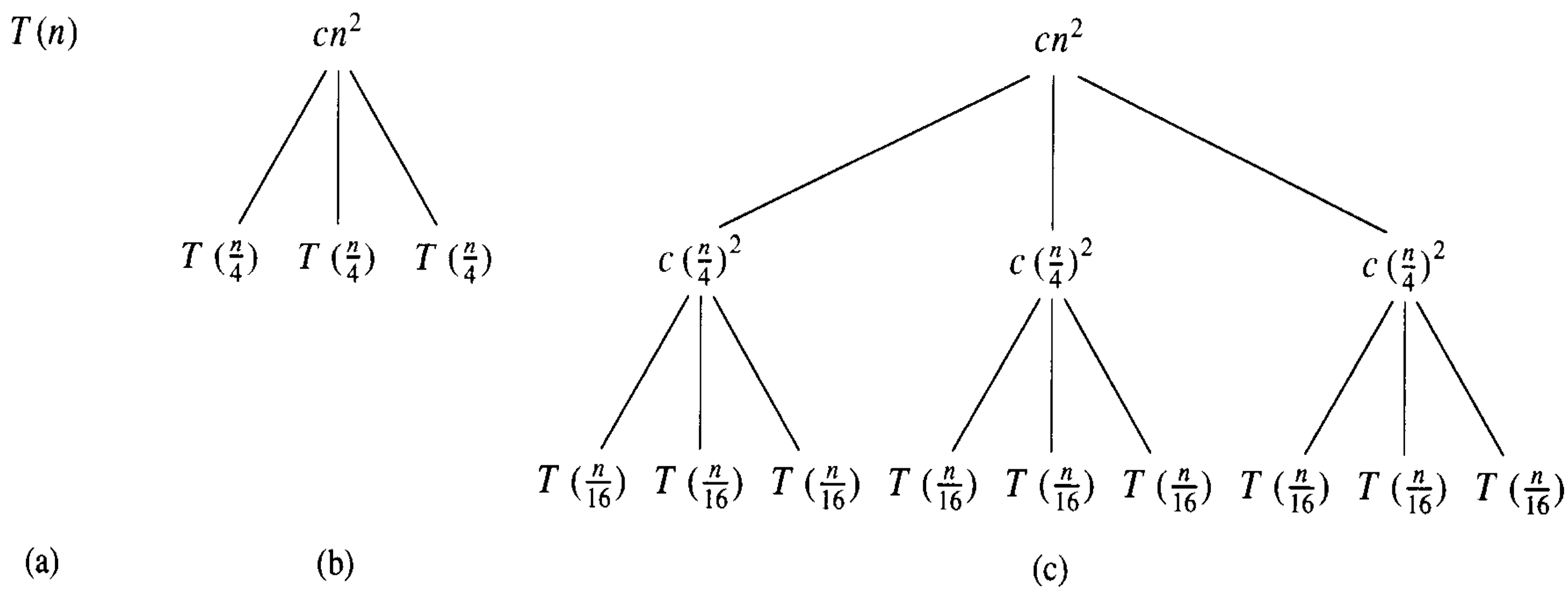
Because we haven't proven the *exact from* of our inductive hypothesis (which is that $T(n) \leq cn$), this proof is false.

Recurrence trees

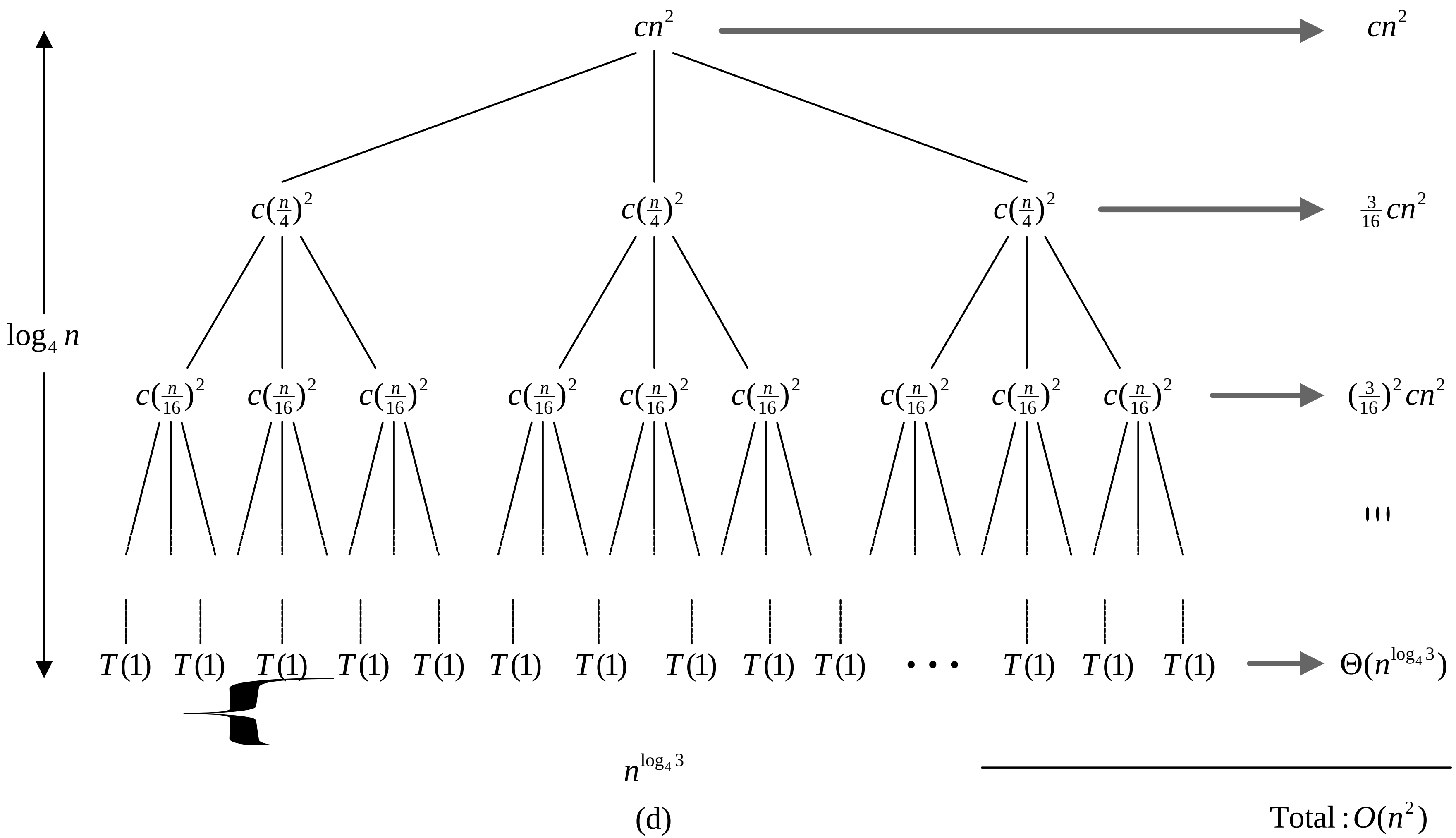
- Goal of the recursion-tree method
  - a good guess for the substitution method
  - a direct proof of a solution to a recurrence (provided by carefully drawing a recursion tree)

Recurrence trees

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



Recurrence trees



(d)

Recurrence trees

- The cost of the entire tree

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

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

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

Recurrence trees

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

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

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

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

$$= O(n^2)$$

Recurrence trees

- Verify by the substitution method
  - Show that $T(n) \le dn^2$ for some constant d > 0

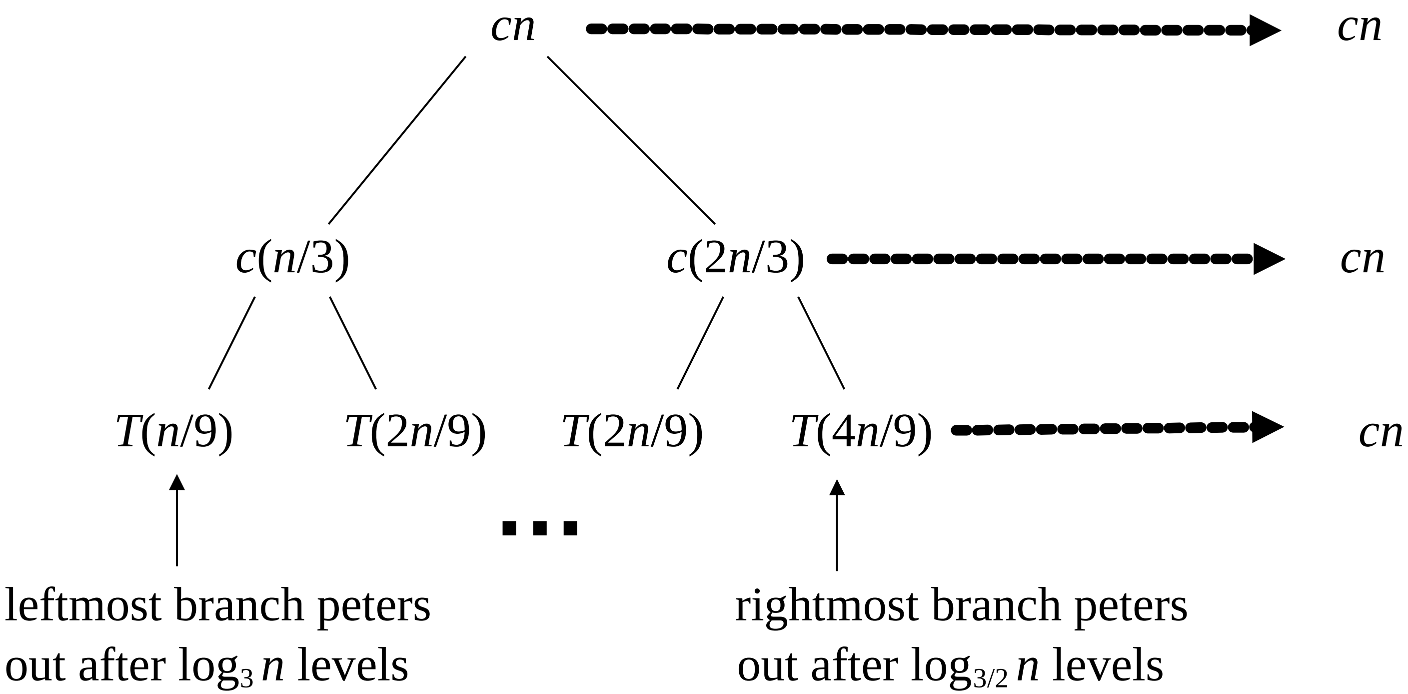
$$
\begin{aligned}
T(n) &\le 3T(\lfloor n/4 \rfloor) + cn^2 \\
&\le 3d\lfloor n/4 \rfloor^2 + cn^2 \\
&\le 3d(n/4)^2 + cn^2 \\
&= \frac{3}{16} dn^2 + cn^2 \\
&\le dn^2,
\end{aligned}
$$

where the last step holds as long as $d \ge (16/13)\,c$

Recurrence trees

Use to generate a guess. Then verify by substitution method.

E.g.: $T(n) = T(n/3) + T(2n/3) + \Theta(n)$. For upper bound, rewrite as $T(n) \leq T(n/3) + T(2n/3) + cn$; for lower bound, as $T(n) \geq T(n/3) + T(2n/3) + cn$. By summing across each level, the recursion tree shows the cost at each level of recursion (minus the costs of recursive calls, which appear in subtrees):



Recurrence trees (cont'd)

- There are $\log_3 n$ full levels, and after $\log_{3/2} n$ levels, the problem size is down to 1.
- Each level contributes $\leq cn$.
- Lower bound guess: $\geq d\, n \log_3 n = \Omega(n \log n)$ for some positive constant d.
- Upper bound guess: $\leq d\, n \log_{3/2} n = O(n \log n)$ for some positive constant d.
- Then *prove* by substitution.

Recurrence trees (cont'd)

1. ***Upper bound:***

   Guess: $T(n) \le dn\lg n$.

   Substitution:

   $T(n) \le T(n/3) + T(2n/3) + cn$

   $\le d\,(n/3)\lg(n/3) + d(2n/3)\lg(2n/3) + cn$

   $= (d\,(n/3)\lg n - d\,(n/3)\lg 3) + (d\,(2n/3)\lg n - d\,(2n/3)\lg(3/2)) + cn$

   $= d\,n\lg n - d\,((n/3)\lg 3 + (2n/3)\lg(3/2)) + cn$

   $= d\,n\lg n - d\,((n/3)\lg 3 + (2n/3)\lg 3 - (2n/3)\lg 2) + cn$

   $= d\,n\lg n - d\,n(\lg 3 - 2/3) + cn$

   $\le d\,n\lg n$ if $-d\,n(\lg 3 - 2/3) + cn \le 0$,

   $$d \ge \frac{c}{\lg 3 - 2/3}$$

   Therefore, $T(n) = O(n\lg n)$.

   Note: Make sure that symbolic constants used in the recurrence (e.g.,c) and the guess (e.g.,d) are different.

Recurrence trees **(cont'd)**

2. ***Lower bound:***

Guess: $T(n) \geq dn \lg n$.

Substitution: Same as for the upper bound, but replacing ≤ by ≥. End up needing

$$0 < d \leq \frac{c}{\lg 3 - 2/3}$$

Therefore, T(n) = Ω(*n* lg *n*).

Since $T(n) = O(n \lg n)$ and $T(n) = \Omega(n \lg n)$, we conclude that $T(n) = \Theta(n \lg n)$

Master method (cond't)

Used for many divide-and-conquer recurrences of the form

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

Where $a \geq 1$, $b > 1$, and $f(n) > 0$.

Based on the ***master theorem*** (Theorem 4.1).

Compare $n^{\log_b a}$ *vs.* $f(n)$*:*

Case 1: $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$.

($f(n)$ is polynomially smaller than $n^{\log_b a}$)

Solution: $T(n) = \Theta(n^{\log_b a})$

Master method (cond't)

Case 2: $f(n) = \Theta(n^{\log_b a} \lg^k n)$, where $k \geq 0$

($f(n)$ is within a polylog factor of $n^{\log_b a}$, *but not smaller*)

Solution: $T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$

Simple case:

$k = 0 \Rightarrow f(n) = \Theta(n^{\log_b a}) \Rightarrow T(n) = \Theta(n^{\log_b a} \lg n)$

Master method (cond't)

Case 3: $f(n) = \Omega(n^{\log_b a+\varepsilon})$ for some constant $\varepsilon > 0$ and $f(n)$ satisfies the regularity condition $a f(n/b) \leq c f(n)$ for some constant $c < 1$ and all sufficiently large n.

($f(n)$ is polynomially greater than $n^{\log_b a}$)

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

Master method (cond't)

***What's with the Case 3 regularity condition*?**

- Generally not a problem.
- It always holds whenever $f(n) = n^k$ and $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for constant $\varepsilon > 0$. So you don't need to check it when $f(n)$ is a polynomial.

Master method (cond't)

Examples:

- $T(n) = 5T(n/2) + \Theta(n^2)$

  $n^{\log_2 5}$ vs. n^2

  Since $\log_2 5 - \varepsilon = 2$ for some constant $\varepsilon > 0$, use Case 1 ➔ $T(n) = \Theta(n^{\lg 5})$

- $T(n) = 27T(n/3) + \Theta(n^3 \lg n)$

  $n^{\log_3 27} = n^3$ vs. $n^3 \lg n$

  Use Case 2 with $k = 1$ ➔ $T(n) = \Theta(n^3 \lg^2 n)$

Master method (cond't)

- $T(n) = 5T(n/2) + \Theta(n^3)$

  $n^{\log_2 5}$ vs. n^3

  Now $\lg 5 + \varepsilon = 3$ for some constant $\varepsilon > 0$

  Check regularity condition (don't really need to since $f(n)$ is a polynomial):

  $a f(n/b) = 5(n/2)^3 = 5n^3/8 \le cn^3$ for $c = 5/8 < 1$

  Use Case 3 ➔ $T(n) = \Theta(n^3)$

- $T(n) = 27T(n/3) + \Theta(n^3/\lg n)$

  $n^{\log_3 27} = n^3$ vs. $n^3/\lg n = n^3 \lg^{-1} n \neq \Theta(n^3 \lg^k n)$ for any $k \ge 0$

  Cannot use the master method.