Lecture Title: Recursion (Cont.)

Dept. of Computer Science
Faculty of Science and Technology

Lecture No:	04	Week No:	04	Semester:	
Lecturer:	*Name & email:*				

Recurrences & Master Method

MD. MANZURUL HASAN

SLIDES ARE ADOPTED FROM MASHIOUR RAHMAN [ASSO DEAN, FST]

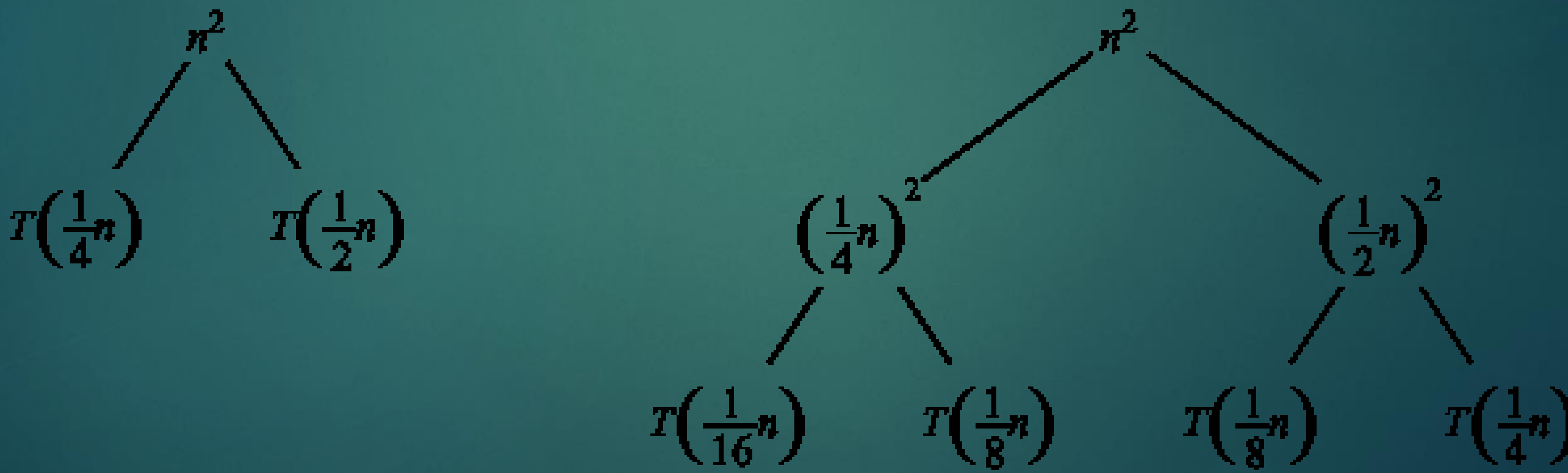
Lecture Outline

1. Divide and Conquer (Previous Week)
2. Recurrences in Divide and Conquer and methodologies for recurrence Solutions (Previous Week)
3. Repeated Backward Substitution Method (Previous Week)
4. Substitution Method (Previous Week)
5. **Recursion Tree**
6. **Master Method**

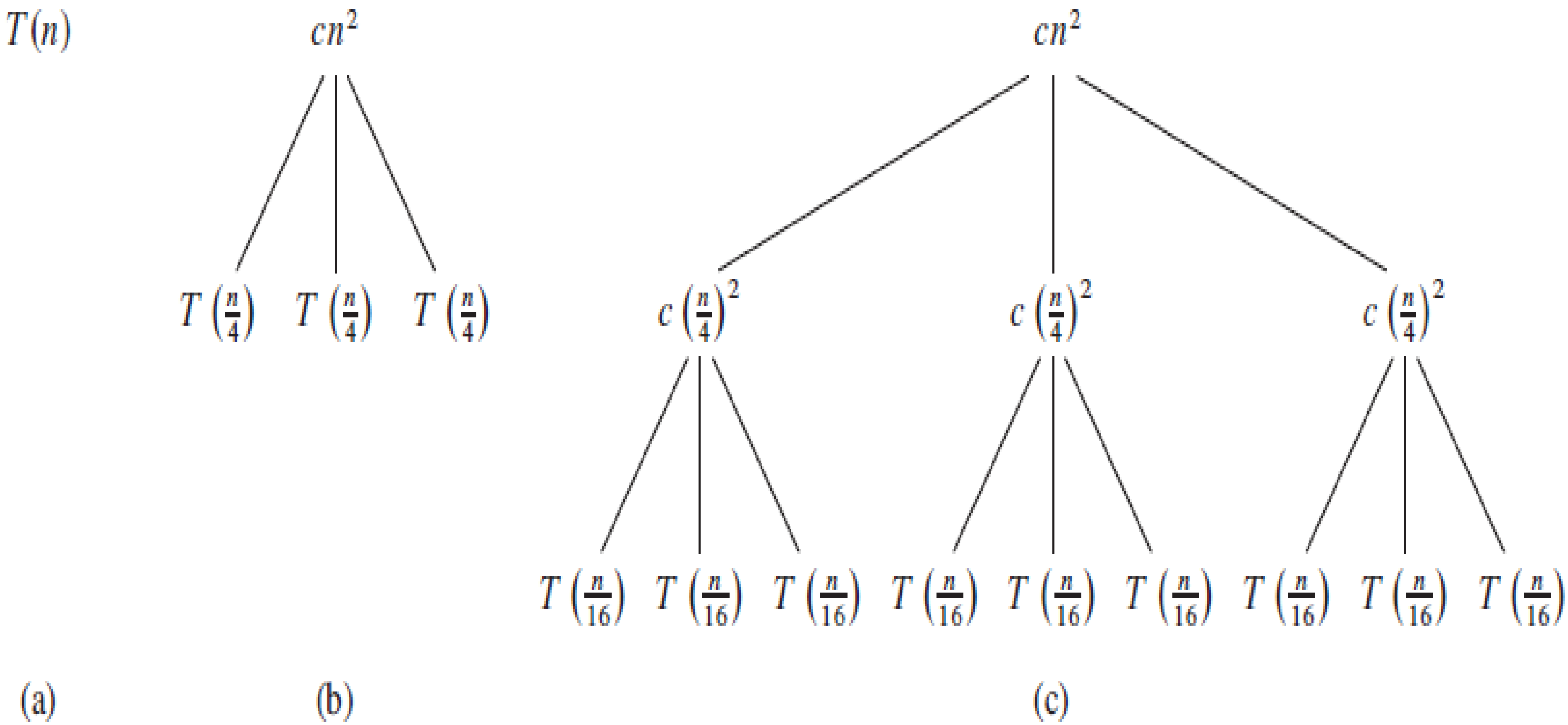
Recursion Tree

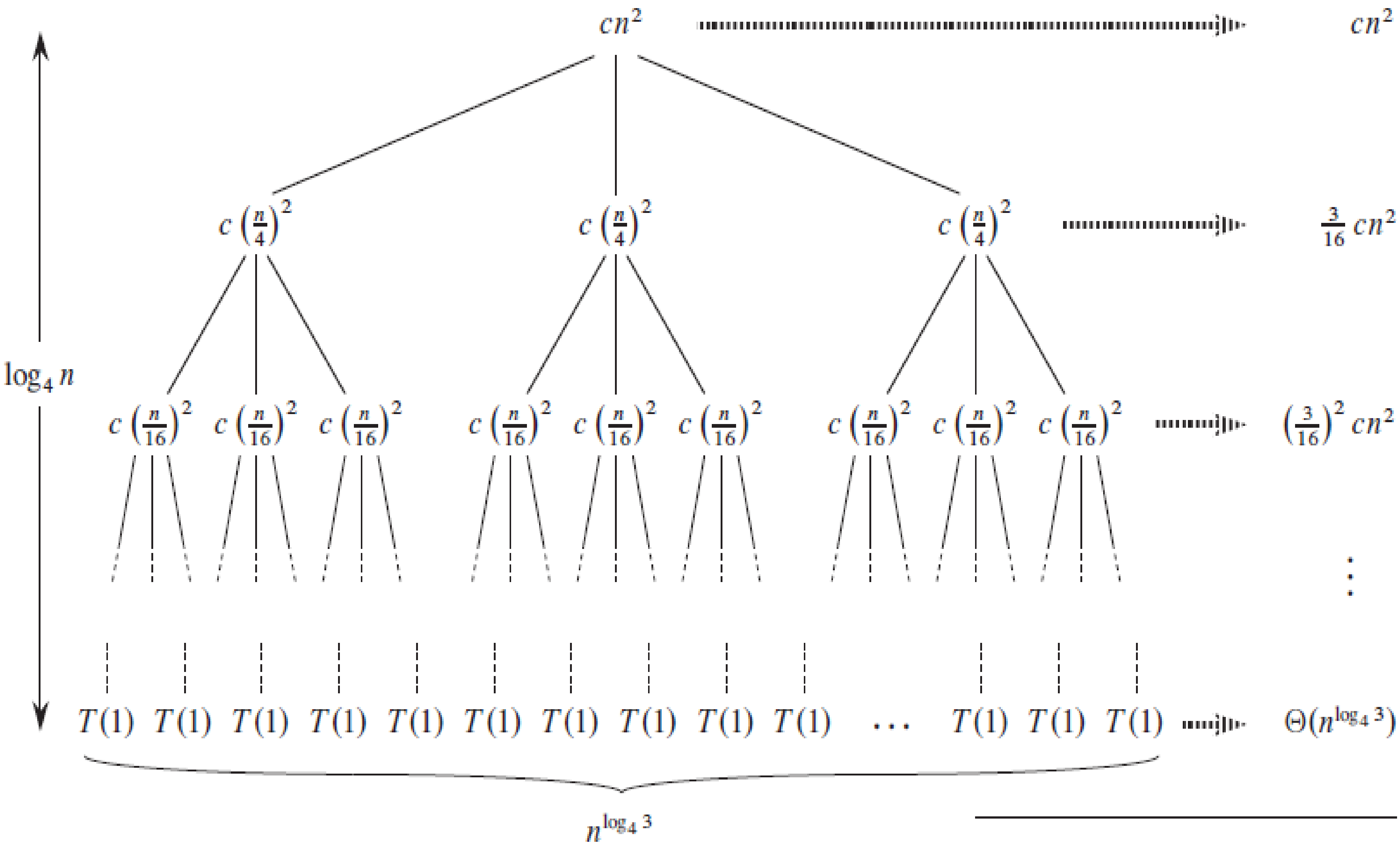
- A recursion tree is a convenient way to visualize what happens when a recurrence is iterated.
  - Good for "guessing" asymtotic solutions to recurrences

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



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





cn^2
$c\left(\frac{n}{4}\right)^2$
$\frac{3}{16}cn^2$
$c\left(\frac{n}{16}\right)^2$
$\left(\frac{3}{16}\right)^2 cn^2$
$\log_4 n$
$T(1)$
$\Theta(n^{\log_4 3})$
$n^{\log_4 3}$
Total: $O(n^2)$

(d)

Height and leaves of a recursion tree

- **Height**
  The top of the tree begins with **n** and for every step down it is divided by **b**. So it goes n, n/b, n/b^2,...,1. To find the height we need to find a **k** such that $n / b^k = 1$ or $\mathbf{b^k = n}$, which gives $\mathbf{k = \log_b n}$.

- **Number of leaves**
  For every step down the tree, the leaves are multiplied by **a** times. The number of leaves is $\mathbf{a^k}$, where k is the number of steps or height of the tree. **The number of leaves =** $\mathbf{a^{\log_b n}}$**.**

It is easy to see that the tree has $a^{\log_b n}$ leaves. Indeed, since the height is $\log_b n$, and the tree branching factor is a, the number of leaves is

$$a^h = a^{\log_b n} = a^{\frac{\log_a n}{\log_a b}} = n^{\frac{1}{\log_a b}} = n^{\log_b a}$$

$$
\begin{aligned}
T(n) &= cn^2 + \frac{3}{16}cn^2 + \left(\frac{3}{16}\right)^2 cn^2 + \cdots + \left(\frac{3}{16}\right)^{\log_4 n-1} cn^2 + \Theta(n^{\log_4 3}) \\
&= \sum_{i=0}^{\log_4 n-1} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\
&= \frac{(3/16)^{\log_4 n} - 1}{(3/16) - 1} cn^2 + \Theta(n^{\log_4 3}) \qquad \text{(by equation (A.5))} .
\end{aligned}
$$

Geometric series

For real $x \neq 1$, the summation

$$\sum_{k=0}^{n} x^k = 1 + x + x^2 + \cdots + x^n$$

is a ***geometric*** or ***exponential series*** and has the value

$$\sum_{k=0}^{n} x^k = \frac{x^{n+1} - 1}{x - 1} . \tag{A.5}$$

When the summation is infinite and $|x| < 1$, we have the infinite decreasing geometric series

$$\sum_{k=0}^{\infty} x^k = \frac{1}{1 - x} . \tag{A.6}$$

$$
\begin{aligned}
T(n) &= \sum_{i=0}^{\log_4 n - 1} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\
&< \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\
&= \frac{1}{1-(3/16)} cn^2 + \Theta(n^{\log_4 3}) \\
&= \frac{16}{13} cn^2 + \Theta(n^{\log_4 3}) \\
&= O(n^2) .
\end{aligned}
$$

Verify our guess [substitution]

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

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

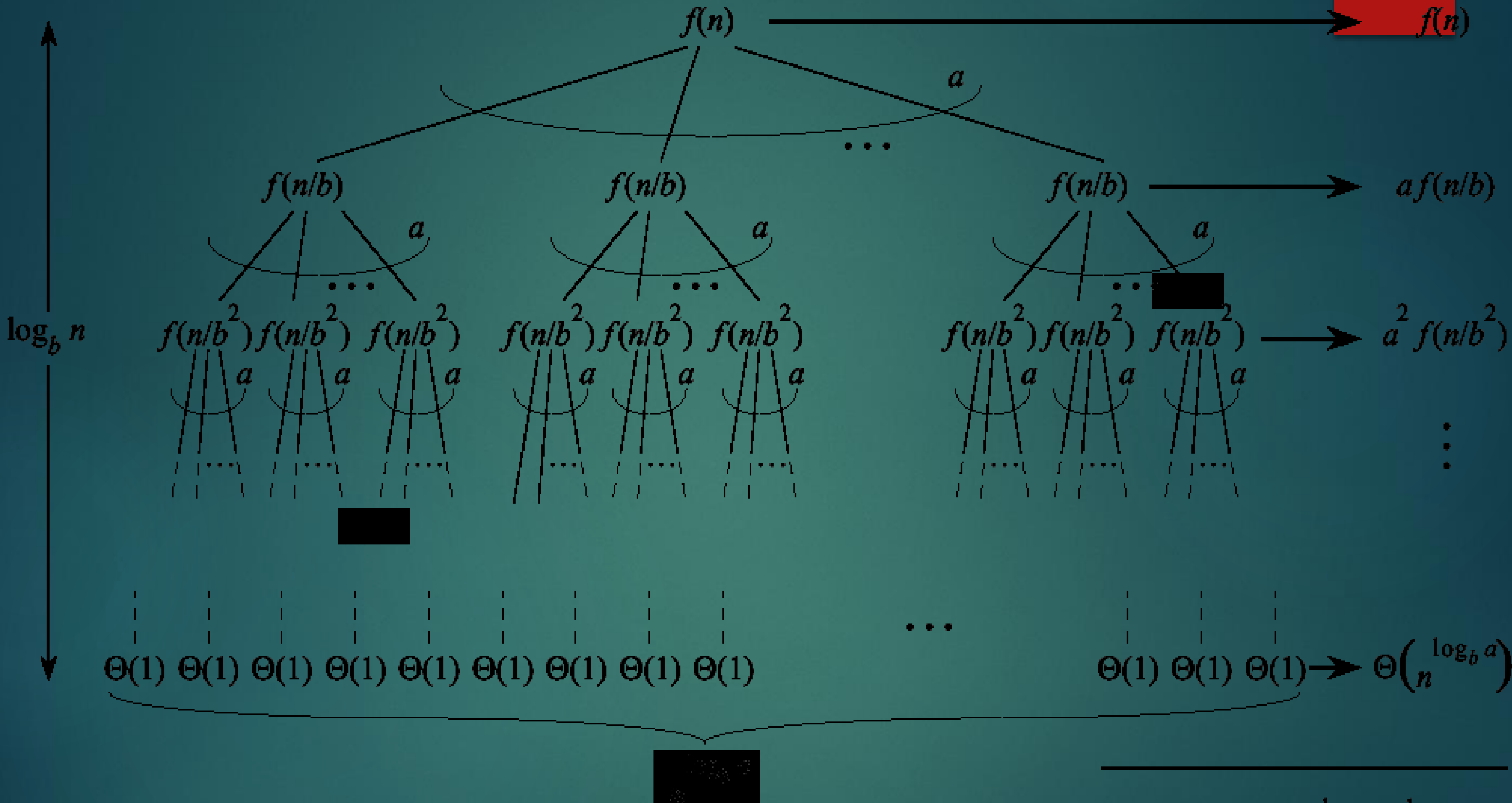
Master Method

- The idea is to solve a class of recurrences that have the form

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

- ***Assumptions:*** ***a*** $\geq$ **1** and ***b*** **> 1**, and ***f(n)*** is asymptotically positive.
- Abstractly speaking, ***T*(*n*)** is the runtime for an algorithm and we know that
  - ***a*** number of subproblems of size ***n/b*** are solved recursively, each in time ***T*(*n/b*).**
  - ***f(n)*** is the cost of dividing the problem and combining the results. e.g., In merge-sort $T(n) = 2T(n/2) + \Theta(n)$

…Master Method



$$\text{Total: } \Theta\left(n^{\log_b a}\right) + \sum_{j=0}^{\log_b n - 1} a^j f(n/b^j)$$

Split problem into a parts. There are $\log_b n$ levels. There are $a^{\log_b n} = n^{\log_b a}$ leaves.

…Master Method

- Iterating the recurrence (expanding the tree) yields

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

$\quad = f(n) + af(n/b) + a^2T(n/b^2)$

$\quad = f(n) + af(n/b) + a^2f(n/b^2) + \ldots$

$\qquad a^{\log_b n-1} f(n/b^{\log_b n-1}) + a^{\log_b n}T(1)$

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

- The first term is a division/recombination cost (totaled across all levels of the tree).
- The second term is the cost of doing all subproblems of size 1 (total of all work pushed to leaves).

Master Method, Intuition

- **Three** common cases:
  1. Running time dominated by cost at leaves.
  2. Running time evenly distributed throughout the tree.
  3. Running time dominated by cost at the root.
- To solve the recurrence, we need to identify the dominant term.
- In each case compare $f(n)$ with $O(n^{\log_b a})$

Master Method, Case 1

- If $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$ then
  - *f(n)* grows polynomially slower than $n^{\log_b a}$ (by factor n^{ε}).
- **The work at the leaf level dominates**

Cost of all the leaves $\Theta(n^{\log_b a})$

Master Method, Case 2

- If $f(n) = \Theta(n^{\log_b a})$ then
  - $f(n)$ and $n^{\log_b a}$ are asymptotically the same
- **The work is distributed equally throughout the tree**

(level cost)(number of levels)

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

Master Method, Case 3

- If $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some constant $\varepsilon > 0$ then
  - Inverse of Case 1
  - *f(n)* grows polynomially faster than $n^{\log_b a}$
  - Also need a "regularity" condition

$$\exists c < 1 \text{ and } n_0 > 0 \text{ such that } af(n/b) \le cf(n) \ \forall n > n_0$$

- **The work at the root dominates**

division/recombination cost

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

Master Theorem, Summarized

Given: recurrence of the form $T(n) = aT(n/b) + f(n)$

1. $f(n) = O\left(n^{\log_b a - \varepsilon}\right)$

   $\Rightarrow T(n) = \Theta\left(n^{\log_b a}\right)$

2. $f(n) = \Theta\left(n^{\log_b a}\right)$

   $\Rightarrow T(n) = \Theta\left(n^{\log_b a} \lg n\right)$

3. $f(n) = \Omega\left(n^{\log_b a + \varepsilon}\right)$ and $af(n/b) \le cf(n)$, for some $c < 1, n > n_0$

   $\Rightarrow T(n) = \Theta\left(f(n)\right)$

Strategy

1. **Extract *a***, ***b***, and ***f(n)*** from a given recurrence
2. **Determine $n^{\log_b a}$**
3. **Compare *f(n)*** and **$n^{\log_b a}$ asymptotically**
4. **Determine** appropriate **MT case and apply it**

Merge sort: $T(n) = 2T(n/2) + \Theta(n)$

1. $a=2, b=2, f(n) = \Theta(n)$

2. $n^{\log_2 2} = n$

3. $\Theta(n) = \Theta(n)$

➔ Case 2: $T(n) = \Theta(n^{\log_b a} \log n) = \Theta(n \log n)$

Examples of Master Method

```
BinarySearch(A, l, r, q):
   m := (l+r)/2
   if A[m]=q then return m
   else if A[m]>q then
        BinarySearch(A, l, m-1, q)
   else BinarySearch(A, m+1, r, q)
```

$T(n) = T(n/2) + 1$

1. $a=1, b=2, f(n) = 1$
2. $n^{\log_2 1} = 1$
3. $1 = \Theta(1)$

➔ Case 2: $T(n) = \Theta(n^{\log_b a} \log n) = \Theta(\log n)$

…Examples of Master Method

$T(n) = 9T(n/3) + n$

1. $a=9, b=3, f(n) = n$
2. $n^{\log_3 9} = n^2$
3. $n = \Theta(n^{\log_3 9 - \varepsilon})$ with $\varepsilon = 1$

➔ Case 1: $T(n) = \Theta(n^2)$

...Examples of Master Method

$T(n) = 3T(n/4) + n \log n$

1. $a=3,\ b=4,\ f(n) = n \log n$

2. $n^{\log_4 3} = n^{0.792}$

3. $n \log n = \Omega(n^{\log_4 3 + \varepsilon})$ with $\varepsilon = 0.208$

➔ **Case 3:**

regularity condition: $af(n/b) <= cf(n)$

$af(n/b) = 3(n/4)\log(n/4) <= (3/4)n\log n = cf(n)$

;with $c=3/4$

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

- $T(n) = 8T(n/2) + n^3$

References & Readings

- CLRS
  - Chapter: 4 (4.1-4.3)
  - 4.4 for bedtime reading
  - Exercises
    - 4.1, 4.2, 4.3
  - Problems
    - 4-1, 4-4
- HSR
  - Chapter: 3 (3.1-3.6)
  - Examples: 3.1-3.5
  - Exercises: 3.1 (1, 2), 3.2 (1, 3-6),