CS11313 - **Spring** 2023

Design & Analysis of Algorithms

Master Method

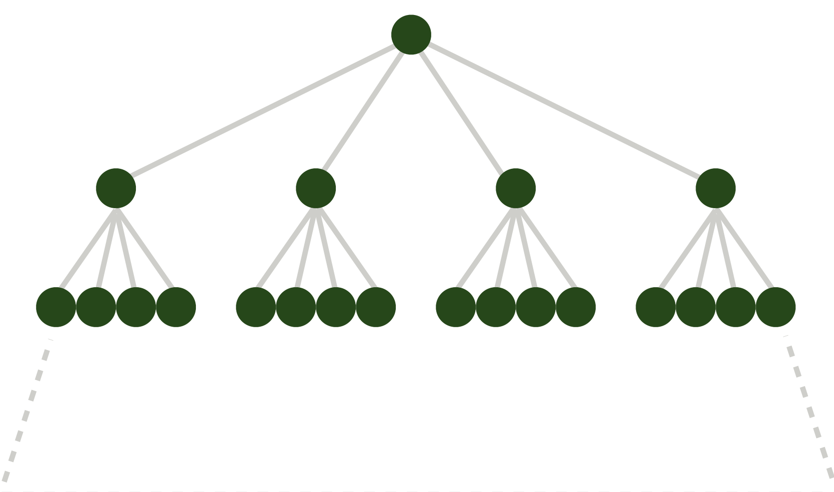
Ibrahim Albluwi

Three Familiar Examples

1

$$T(n) = \begin{cases} 4T(\frac{n}{2}) + n & \text{if } n > 1 \\ 1 & \text{if } n \le 1 \end{cases}$$

work at the **root** = n



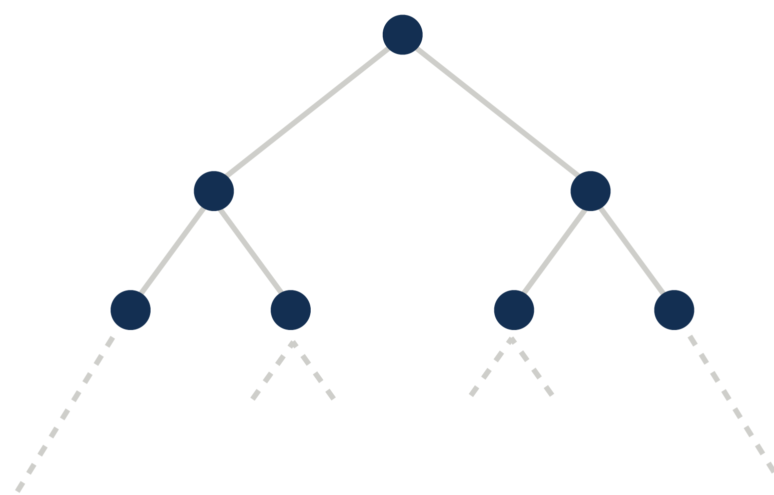
number of **leaves** = $4^{\log_2 n} = n^2$

$$T(n) = \sum_{i=0}^{\log_2 n} 4^i (\frac{n}{2^i}) = \Theta(n^2)$$

2

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

work at the **root** = n



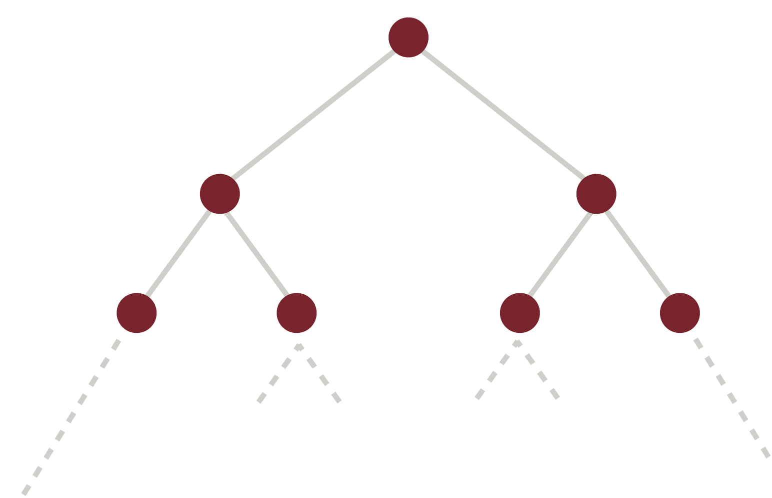
number of **leaves** = $2^{\log_2 n} = n$

$$T(n) = \sum_{i=0}^{\log_2 n} 2^i (\frac{n}{2^i}) = \Theta(n \log n)$$

3

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

work at the **root** = n^2



number of **leaves** = $2^{\log_2 n} = n$

$$T(n) = \sum_{i=0}^{\log_2 n} 2^i (\frac{n}{2^i})^2 = \Theta(n^2)$$

Claim. If # of leaves $\succ$ work at the root: $T(n) = \Theta(\text{number leaves})$ — tree is **leaf** dominated

If # of leaves $\equiv$ work at the root: $T(n) = \Theta(\text{work at the root} \times \text{number of levels})$ — all levels are the same

If # of leaves $\prec$ work at the root: $T(n) = \Theta(\text{work at the root})$ — tree is **root** dominated

Another Three Familiar Examples

1

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

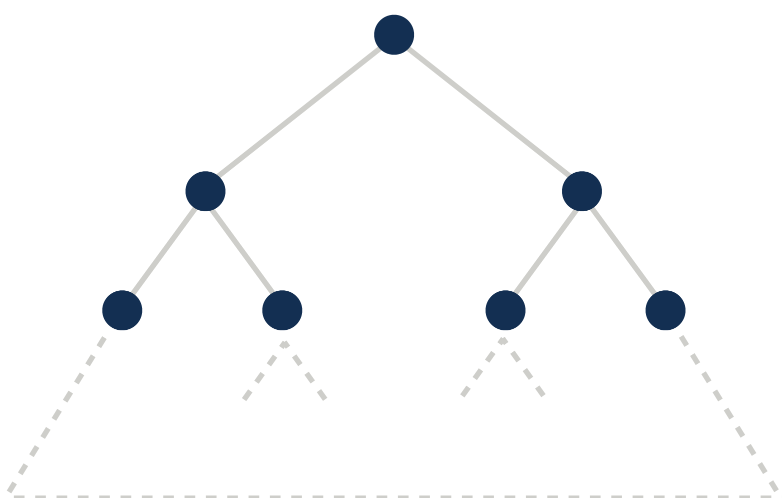
2

$$T(n) = \begin{cases} 3T(\frac{n}{9}) + \sqrt{n} & \text{if } n > 1 \\ 1 & \text{if } n \le 1 \end{cases}$$

3

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

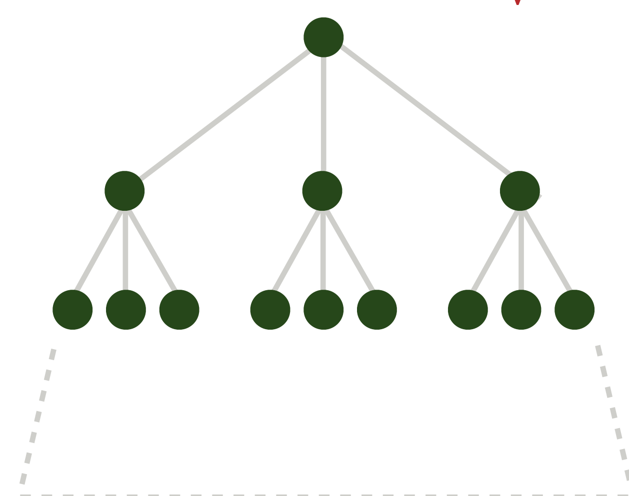
work at the **root** = c



number of **leaves** = $2^{\log_2 n} = n$

$$T(n) \;=\; c \times \sum_{i=0}^{\log_2 n} 2^i \;=\; \Theta(n)$$

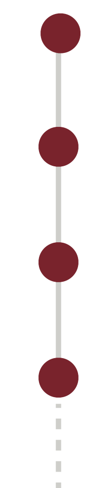
work at the **root** = $\sqrt{n}$



number of **leaves** = $3^{\log_9 n} = \sqrt{n}$

$$T(n) \;=\; \sum_{i=0}^{\log_9 n} 3^i \sqrt{\frac{n}{9^i}} \;=\; \Theta(\sqrt{n}\log n)$$

work at the **root** = n



number of **leaves** = 1

$$T(n) \;=\; \sum_{i=0}^{\log_2 n} \frac{n}{2^i} \;=\; \Theta(n)$$

Claim. If # of leaves $\succ$ work at the root: $T(n) = \Theta(\text{number leaves})$ — tree is **leaf** dominated

If # of leaves $\equiv$ work at the root: $T(n) = \Theta(\text{work at the root} \times \text{number of levels})$ — all levels are the same

If # of leaves $\prec$ work at the root: $T(n) = \Theta(\text{work at the root})$ — tree is **root** dominated

Master Method

Given a recurrence equation of the following form:

$$T(n) = \begin{cases} aT(\frac{n}{b}) + f(n) & \text{if } n > 1 \\ \Theta(1) & \text{if } n = 1 \end{cases}$$

Where n is a positive integer, $a \geq 1$ and $b > 1$, then:

there is *at least one* whole subproblem!

subproblems *decrease* in size

Master Method

Given a recurrence equation of the following form:

$$T(n) = \begin{cases} aT(\frac{n}{b}) \ + \ f(n) & \text{if } \ n > 1 \\ \Theta(1) & \text{if } \ n = 1 \end{cases}$$

Where n is a positive integer, $a \geq 1$ and $b > 1$, then:

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

Informally. If the work at the root is *polynomially less* than the number of leaves:

$T(n) = \Theta(\text{number of leaves})$

Example. $T(n) = 4T(\frac{n}{2}) + n \log n$

$f(n) = n \log n$

$n \log n = O(n^{\log_2 4 - \epsilon})$

$= O(n^{2-\epsilon})$ for all $\epsilon < 1$

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

Example. $T(n) = 2T(\frac{n}{2}) + n \log n$

$f(n) = n \log n$

$n \log n \neq O(n^{\log_2 2 - \epsilon})$

$\neq O(n^{1-\epsilon})$

Case 1 does not apply!

$f(n)$ = work at the root | Number of leaves $= a^{\log_b n} = n^{\log_b a}$

Master Method

Given a recurrence equation of the following form:

$$T(n) = \begin{cases} aT(\frac{n}{b}) + f(n) & \text{if } n > 1 \\ \Theta(1) & \text{if } n = 1 \end{cases}$$

Where n is a positive integer, $a \geq 1$ and $b > 1$, then:

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

Case 2. If $f(n) = \Theta(n^{\log_b a})$ then $T(n) = \Theta(f(n) \bullet \log n)$

Informally. If the work at the root is *asymptotically the same* as the number of leaves:

$T(n) = \Theta(\text{work at the root} \times \text{number of levels})$

Example. $T(n) = 2T(\frac{n}{2}) + n$

$f(n) = n = \Theta(n^{\log_2 2})$

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

Example. $T(n) = 2T(\frac{n}{2}) + n \log n$

$f(n) = n \log n \neq \Theta(n^{\log_2 2})$

Case 2 does not apply

$f(n)$ = work at the root | Number of leaves $= a^{\log_b n} = n^{\log_b a}$

Master Method

Given a recurrence equation of the following form:

$$T(n) = \begin{cases} aT(\frac{n}{b}) + f(n) & \text{if } n > 1 \\ \Theta(1) & \text{if } n = 1 \end{cases}$$

Where n is a positive integer, $a \geq 1$ and $b > 1$, then:

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

Case 2. If $f(n) = \Theta(n^{\log_b a})$ then $T(n) = \Theta(f(n) \bullet \log n)$

Case 3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ then $T(n) = \Theta(f(n))$ (for some constant $\epsilon > 0$)

Informally. If work at the root is *polynomially greater* than the # of leaves: $T(n) = \Theta(\text{work at the root})$

Example. $T(n) = 2T(\frac{n}{2}) + n^2$

$f(n) = n^2 = \Omega(n^{\log_2 2 + \epsilon}) = \Omega(n^{1+\epsilon})$

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

Example. $T(n) = 2T(\frac{n}{2}) + n \log n$

$f(n) = n \log n \neq \Omega(n^{1+\epsilon})$

Case 3 does not apply

$f(n)$ = work at the root | Number of leaves $= a^{\log_b n} = n^{\log_b a}$

Master Method

Given a recurrence equation of the following form:

$$T(n) = \begin{cases} aT(\frac{n}{b}) + f(n) & \text{if } n > 1 \\ \Theta(1) & \text{if } n = 1 \end{cases}$$

Where n is a positive integer, $a \geq 1$ and $b > 1$, then:

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

Case 2. If $f(n) = \Theta(n^{\log_b a})$ then $T(n) = \Theta(f(n) \bullet \log n)$

Case 3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ then $T(n) = \Theta(f(n))$ (for some constant $\epsilon > 0$)

provided that there are constants $c < 1$ and n_0, such that $af(\frac{n}{b}) \leq cf(n)$ for all $n \geq n_0$.

Regularity Condition: work at children $\leq$ work at the parent

$f(n)$ = work at the root | Number of leaves $= a^{\log_b n} = n^{\log_b a}$

Case 1 (Examples)

Given a recurrence equation of the following form:

$$T(n) = \begin{cases} aT(\frac{n}{b}) \; + \; f(n) & \text{if } \; n > 1 \\ \Theta(1) & \text{if } \; n = 1 \end{cases}$$

Where n is a positive integer, $a \geq 1$ and $b > 1$, then:

Case 1 (Tree is leaf-dominated)

If $f(n) = O(n^{\log_b a - \epsilon})$ **then**

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

(for some constant $\epsilon > 0$)

Recurrence	$f(n)$	# of leaves	Case 1 condition	Result
$T(n) = 4T(\frac{n}{2}) + n$				
$T(n) = 2T(\frac{n}{2}) + c$				
$T(n) = 3T(\frac{n}{2}) + \sqrt{n}$				

$f(n)$ = work at the root | Number of leaves $= a^{\log_b n} = n^{\log_b a}$

Case 1 (Examples)

Given a recurrence equation of the following form:

$$T(n) = \begin{cases} aT(\frac{n}{b}) \; + \; f(n) & \text{if } \; n > 1 \\ \Theta(1) & \text{if } \; n = 1 \end{cases}$$

Where n is a positive integer, $a \geq 1$ and $b > 1$, then:

Case 1 (Tree is leaf-dominated)

If $f(n) = O(n^{\log_b a - \epsilon})$ **then**

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

(for some constant $\epsilon > 0$)

Recurrence	$f(n)$	# of leaves	Case 1 condition	Result
$T(n) = 4T(\frac{n}{2}) + n$	n	$n^{\log_2 4} = n^2$	$n = O(n^{2-\epsilon})$ If we pick $\epsilon \leq 1$	$T(n) = \Theta(n^2)$
$T(n) = 2T(\frac{n}{2}) + c$	c	$n^{\log_2 2} = n^1$	$c = O(n^{1-\epsilon})$ If we pick $\epsilon \leq 1$	$T(n) = \Theta(n)$
$T(n) = 3T(\frac{n}{2}) + \sqrt{n}$	$\sqrt{n}$	$n^{\log_2 3} = n^{1.585}$	$n^{0.5} = O(n^{1.585-\epsilon})$ If we pick $\epsilon \leq 1.085$	$T(n) = \Theta(n^{1.585})$

$f(n)$ = work at the root | Number of leaves $= a^{\log_b n} = n^{\log_b a}$

Case 3 (Examples)

Given a recurrence equation of the following form:

$$T(n) = \begin{cases} aT(\frac{n}{b}) + f(n) & \text{if } n > 1 \\ \Theta(1) & \text{if } n = 1 \end{cases}$$

Where n is a positive integer, $a \geq 1$ and $b > 1$, then:

Case 3 (Tree is root-dominated)

If $f(n) = \Omega(n^{\log_b a + \epsilon})$ **then**

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

(for some constant $\epsilon > 0$)

Regularity Condition: $af(\frac{n}{b}) \leq cf(n)$ for some $c < 1$

Recurrence	# of leaves	Case 3 condition	$af(\frac{n}{b}) \leq cf(n)$	Result
$T(n) = 2T(\frac{n}{2}) + n^2$				
$T(n) = T(\frac{n}{2}) + n$				
$T(n) = T(\frac{n}{2}) + \log_2 n$				

$f(n)$ = work at the root | Number of leaves $= a^{\log_b n} = n^{\log_b a}$

Case 3 (Examples)

Given a recurrence equation of the following form:

$$T(n) = \begin{cases} aT(\frac{n}{b}) \ + \ f(n) & \text{if } \ n > 1 \\ \Theta(1) & \text{if } \ n = 1 \end{cases}$$

Where n is a positive integer, $a \geq 1$ and $b > 1$, then:

Case 3 (Tree is root-dominated)

If $f(n) = \Omega(n^{\log_b a + \epsilon})$ **then**

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

(for some constant $\epsilon > 0$)

Regularity Condition: $af(\frac{n}{b}) \leq cf(n)$ for some $c < 1$

Recurrence	# of leaves	Case 3 condition	$af(\frac{n}{b}) \leq cf(n)$	Result
$T(n) = 2T(\frac{n}{2}) + n^2$	$n^{\log_2 2} = n^1$	$n^2 = \Omega(n^{1+\epsilon})$ If we pick $\epsilon \leq 1$	$2 \bullet (\frac{n}{2})^2 \leq c \bullet n^2$ $\frac{1}{2}n^2 \leq c \bullet n^2$ pick $0.5 < c < 1$	$T(n) = \Theta(n^2)$
$T(n) = T(\frac{n}{2}) + n$	1	$n = \Omega(n^{0+\epsilon})$ If we pick $\epsilon \leq 1$	$1 \bullet (\frac{n}{2}) \leq c \bullet n$ $\frac{1}{2}n \leq c \bullet n$ pick $0.5 \leq c < 1$	$T(n) = \Theta(n)$
$T(n) = T(\frac{n}{2}) + \log_2 n$	$n^{\log_2 1} = n^0$	$\log_2 n \neq \Omega(n^{0+\epsilon})$		FAIL

$f(n)$ = work at the root | Number of leaves $= a^{\log_b n} = n^{\log_b a}$

Exercises

1. $T(n) = 3T(\frac{n}{2}) + n\sqrt{n}$

2. $T(n) = 2T(\frac{n}{2}) + \log_2 n$

3. $T(n) = T(\frac{n}{2}) + c$

4. $T(n) = T(\frac{n}{2}) + n \log_2 n$

Exercises

1. $T(n) = 3T(\frac{n}{2}) + n\sqrt{n}$ $\quad f(n) = n\sqrt{n},\quad a = 3,\ b = 2,\quad$ # of leaves = $n^{\log_2 3} = n^{1.585}$

$n^{1.5} = O(n^{1.585-\epsilon})$ if we pick $\epsilon \leq 0.085$, Therefore **Case 1** applies: $T(n) = \Theta(n^{1.585})$

2. $T(n) = 2T(\frac{n}{2}) + \log_2 n$ $\quad f(n) = \log_2 n,\quad a = 2,\ b = 2,\quad$ # of leaves = $n^{\log_2 2} = n^1$

$\log_2 n = O(n^{1-\epsilon})$ if we pick $\epsilon < 1$, Therefore **Case 1** applies: $T(n) = \Theta(n)$

3. $T(n) = T(\frac{n}{2}) + c$ $\quad f(n) = c,\quad a = 1,\ b = 2,\quad$ # of leaves = $n^{\log_2 1} = n^0 = 1$

$f(n) = \Theta(n^{\log_b a})$. Therefore, **Case 2** applies: $T(n) = \Theta(c \times \log n)$

4. $T(n) = T(\frac{n}{2}) + n\log_2 n$ $\quad f(n) = n\log_2 n,\quad a = 1,\ b = 2,\quad$ # of leaves = $n^{\log_2 1} = n^0 = 1$

$n\log_2 n = \Omega(n^{0+\epsilon})$, **Case 3** might apply. Check the regularity condition: $af(\frac{n}{b}) \leq cf(n)$.
$1 \bullet \frac{n}{2}\log_2\frac{n}{2} \leq c \bullet n\log_2 n$ is true. Therefore, **Case 3** applies: $T(n) = \Theta(n\log n)$

Examples for **Cases** Where the **Master Method** does not **Apply**

1. $T(n) = T(\frac{n}{2}) + T(\frac{n}{3}) + \Theta(n)$ — Subproblems are not of an equal size.

2. $T(n) = 2T(n-1) + \Theta(n)$ — Subproblems decrease linearly in size.

3. $T(n) = \frac{1}{2}T(\frac{n}{2}) + \Theta(n)$ — Number of subproblems is less than 1.

4. $T(n) = nT(\frac{n}{2}) + \Theta(n)$ — Number of subproblems is not constant.

5. $T(n) = 2T(\frac{n}{2}) - n$ — $f(n)$ is not positive

6. $T(n) = 2T(\frac{n}{2}) + \Theta(n \log n)$ — No polynomial separation between $f(n)$ and the number of leaves.

7. $T(n) = T(\frac{n}{2}) + n(2 \cos n)$ — Regularity condition does not hold.
   There is no constant c for which
   $\frac{n}{2}(2\cos(\frac{n}{2})) \leq cn(2\cos n)$ is always true for large n.