Lecture 5:
Solving Recurrences via the Master Method

These slides include material originally prepared by Dr. Ron Cytron, Dr. Jeremy Buhler, and Dr. Steve Cole.

Announcements

- Exam 1 tomorrow night (see Piazza post for all details)
    - Crib sheet, ID, where to go
- Lab 1 grades posted, Lab 3 grades in progress
    - 1 week regrade request deadline from posting time
- Studio 5 on Thursday as normal

Overview: recurrence-solving strategies

- **Problem: given a recurrence for T(n), find a closed-form asymptotic complexity function that satisfies the recurrence.**

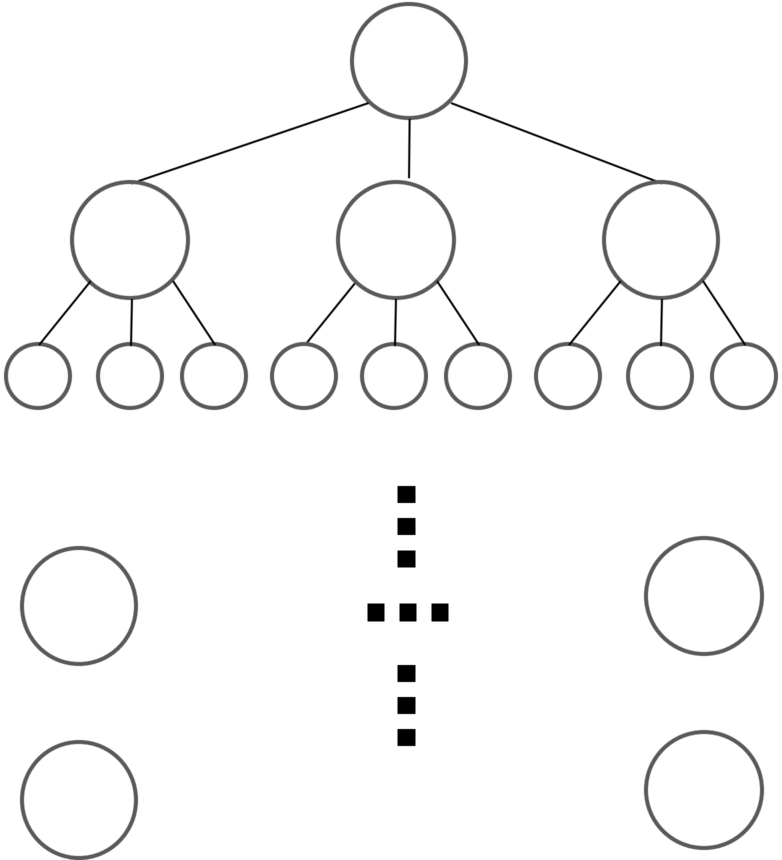
- Possible strategies
  - Guess and check (a.k.a. substitution)
  - Recursion tree accounting (for certain kinds of recurrence)
  - **Master Method (for certain kinds of recurrence)**

Example: $T(n) = 3T(n/4) + cn^2$ $[T(1) = d]$

- [The same one we did at the end of last time]

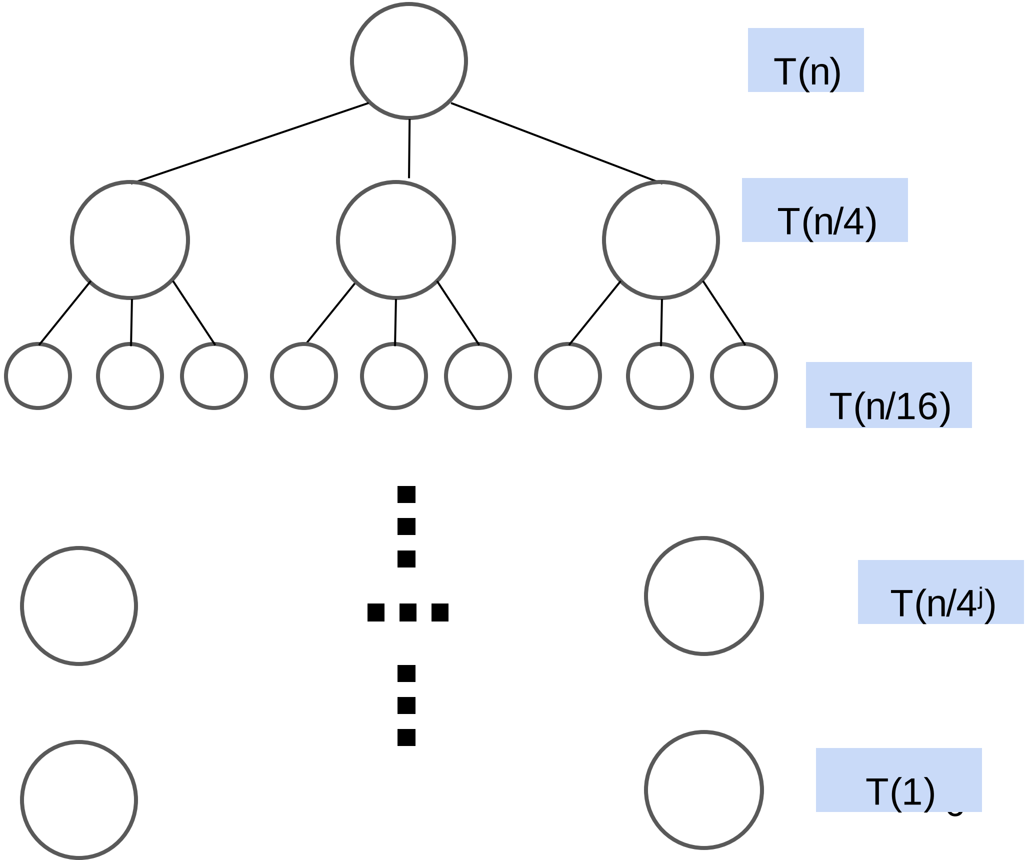
Example: T(n) = 3T(n/4) + cn^2 [T(1) = d]

- This time, a = 3, so each node branches 3 ways!



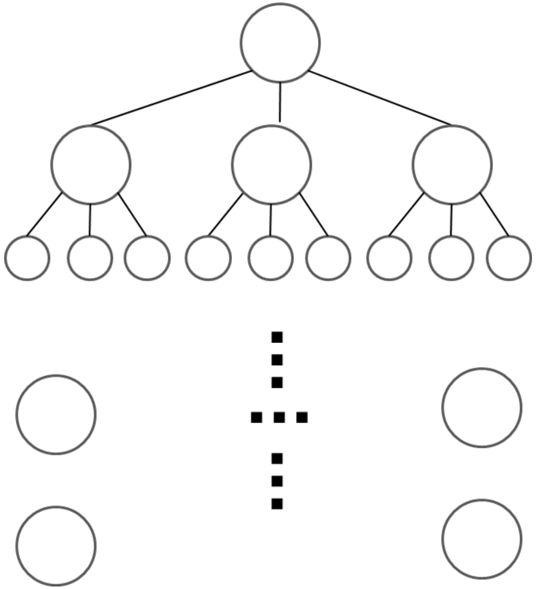
Example: $T(n) = 3T(n/4) + cn^2$ [$T(1) = d$]

- This time, a = 3, so each node branches 3 ways!
- This time, b = 4, so problem size goes down by factor of 4 per level.



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

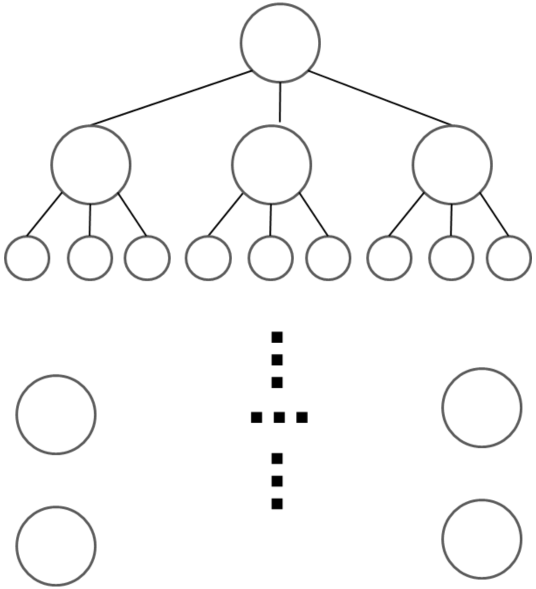
$$T(1) = d$$



Depth	Problem Size	# Nodes Per Level	Local Work per Node
0	n		
1	n/4		
2	n/16		
j	$n/4^j$		
???	1		

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

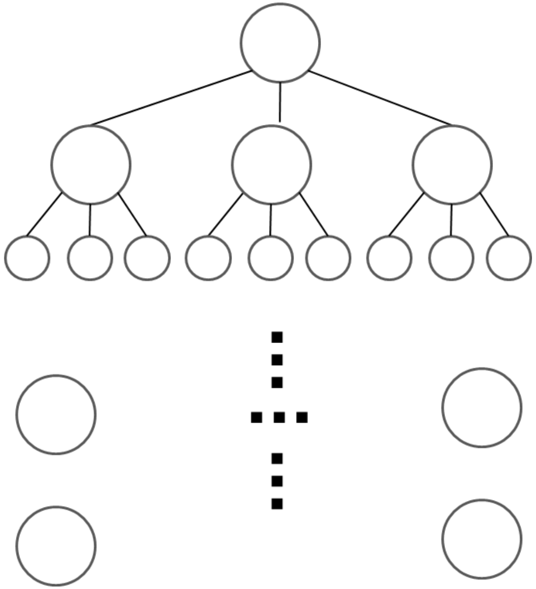
$T(1) = d$



Depth	Problem Size	# Nodes Per Level	Local Work per Node
0	n	1	
1	n/4	3	
2	n/16	9	
j	$n/4^j$	3^j	
$\log_4 n$	1	???	

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

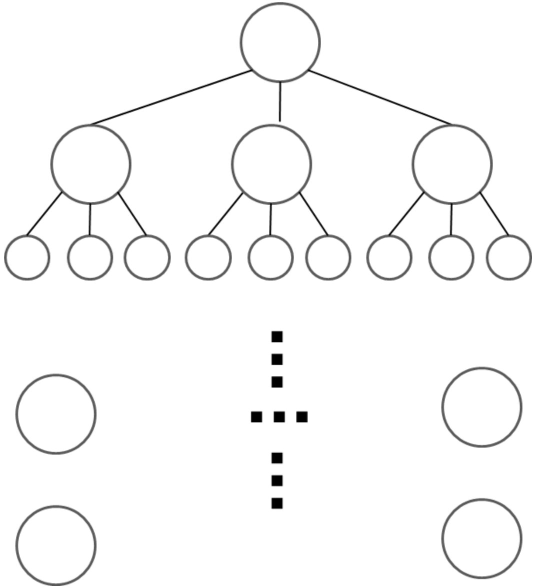
$T(1) = d$



Depth	Problem Size	# Nodes Per Level	Local Work per Node
0	n	1	
1	n/4	3	
2	n/16	9	
j	$n/4^j$	3^j	
$\log_4 n$	1	$3^{\log_4 n}$	

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

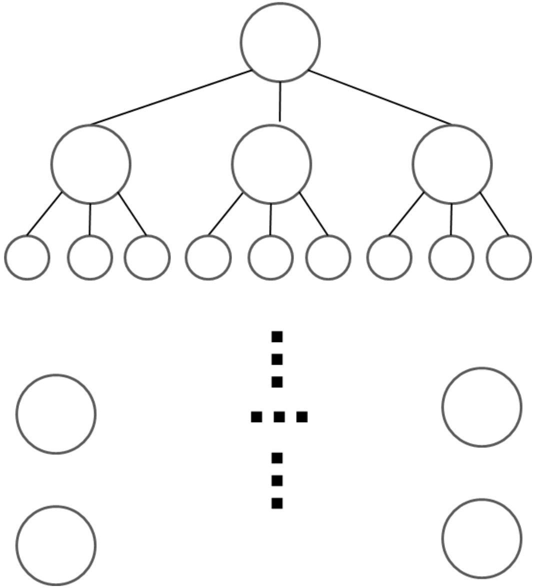
$T(1) = d$



Depth	Problem Size	# Nodes Per Level	Local Work per Node
0	n	1	
1	n/4	3	
2	n/16	9	
j	$n/4^j$	3^j	
$\log_4 n$	1	$n^{\log_4 3}$	

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

$$T(1) = d$$



Depth	Problem Size	# Nodes Per Level	Local Work per Node
0	n	1	cn^2
1	n/4	3	$c(n/4)^2$
2	n/16	9	$c(n/16)^2$
j	$n/4^j$	3^j	$c(n/4^j)^2$
$\log_4 n$	1	$n^{\log_4 3}$	d

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

Depth	Problem Size	# Nodes Per Level	Local Work per Node	Local Work per Level
0	n	1	cn^2	
1	n/4	3	$c(n/4)^2$	
2	n/16	9	$c(n/16)^2$	
j	$n/4^j$	3^j	$c(n/4^j)^2$	
$\log_4 n$	1	$n^{\log_4 3}$	d	

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

Depth	Problem Size	# Nodes Per Level	Local Work per Node	Local Work per Level
0	n	1	cn^2	$1 \text{ x } cn^2$
1	n/4	3	$c(n/4)^2$	$3 \text{ x } c(n/4)^2$
2	n/16	9	$c(n/16)^2$	$9 \text{ x } c(n/16)^2$
j	$n/4^j$	3^j	$c(n/4^j)^2$	$3^j \text{ x } c(n/4^j)^2$
$\log_4 n$	1	$n^{\log_4 3}$	d	$dn^{\log_4 3}$

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

Depth	Problem Size	# Nodes Per Level	Local Work per Node	Local Work per Level
0	n	1	cn^2	cn^2
1	n/4	3	$c(n/4)^2$	$3c(n/4)^2$
2	n/16	9	$c(n/16)^2$	$9c(n/16)^2$
j	$n/4^j$	3^j	$c(n/4^j)^2$	$3^j c(n/4^j)^2$
$\log_4 n$	1	$n^{\log_4 3}$	d	$dn^{\log_4 3}$

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

Depth	Problem Size	# Nodes Per Level	Local Work per Node	Local Work per Level
0				
1				
2				
j				
$\log_4 n$	1	$n^{\log_4 3}$	d	$dn^{\log_4 3}$

$$T(n) = dn^{\log_4 3} + \sum_{j=0}^{\log_4 n - 1} 3^j c\left(n/4^j\right)^2$$

Let’s Break This Summation Down a Bit

$$T(n) = dn^{\log_4 3} + \sum_{j=0}^{\log_4 n - 1} 3^j c\left(n/4^j\right)^2$$

Let’s Break This Summation Down a Bit

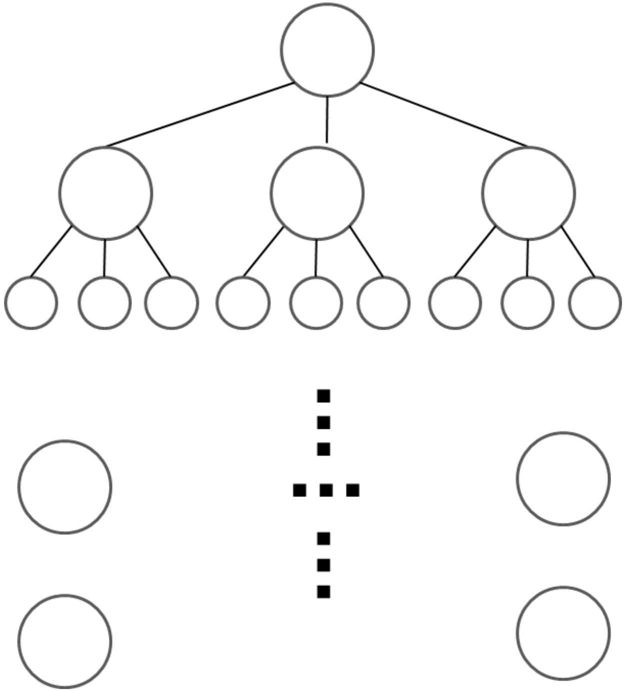
$$T(n) = dn^{\log_4 3} + cn^2 + \sum_{j=1}^{\log_4 n-1} 3^j c\left(n/4^j\right)^2$$

(pulled first term out of sum)

T(n) = 3T(n/4) + cn²

T(1) = d

$$T(n) = dn^{\log_4 3} + cn^2 + \sum_{j=1}^{\log_4 n - 1} 3^j c\left(n/4^j\right)^2$$

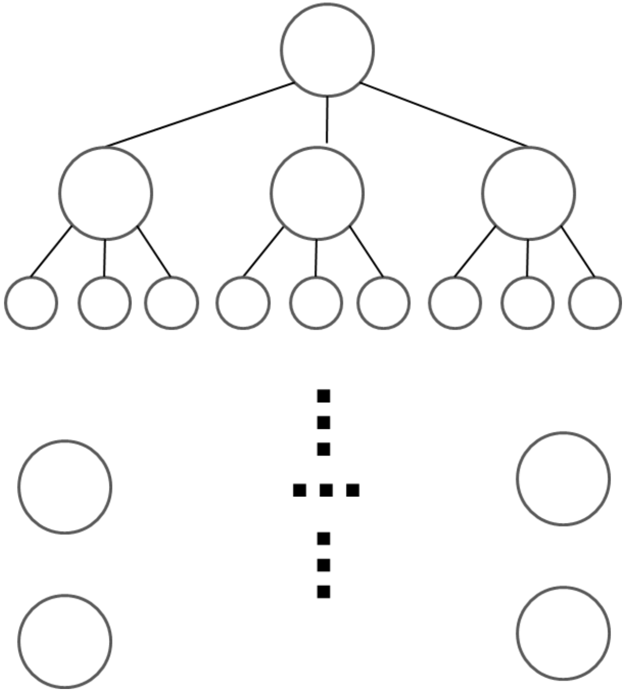


Which parts of the tree contribute to which parts of the sum?

T(n) = 3T(n/4) + cn^2

T(1) = d

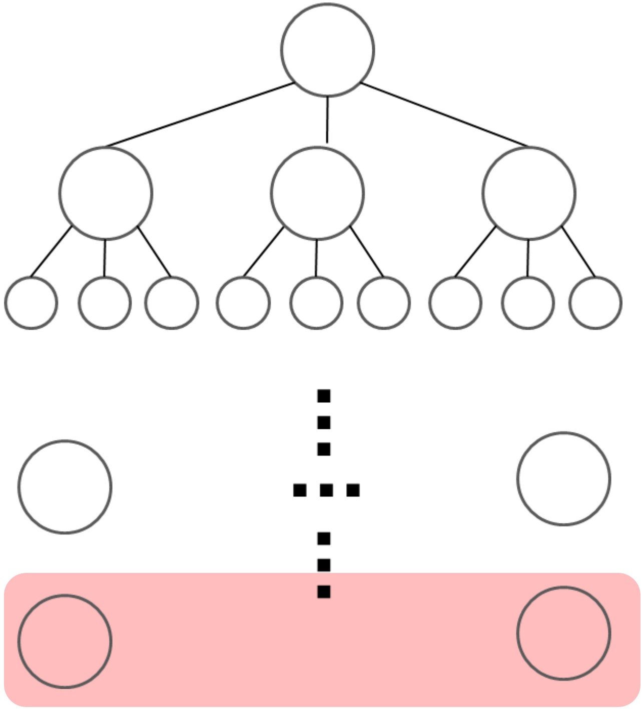
$$T(n) = \boxed{dn^{\log_4 3}} + cn^2 + \sum_{j=1}^{\log_4 n - 1} 3^j c\left(n/4^j\right)^2$$



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

$T(1) = d$

$$T(n) = dn^{\log_4 3} + cn^2 + \sum_{j=1}^{\log_4 n - 1} 3^j c\left(n/4^j\right)^2$$

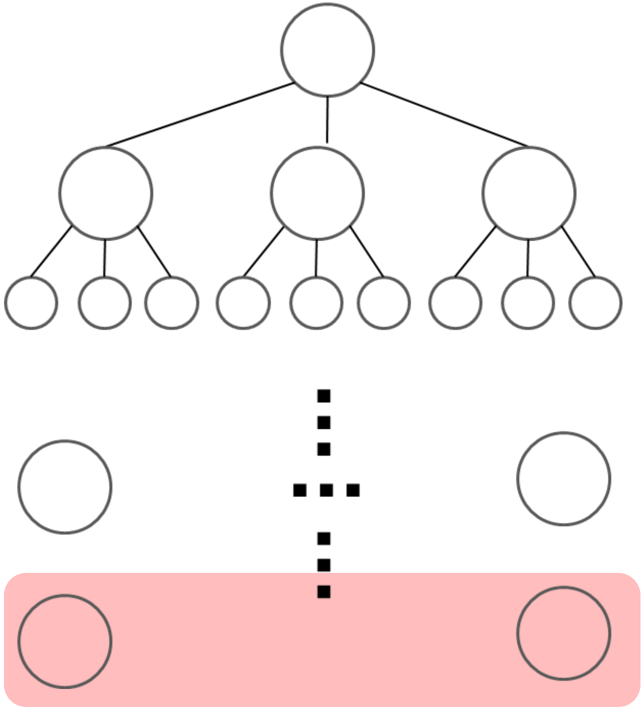


This term is from the *base case* (i.e. bottom of the tree).

T(n) = 3T(n/4) + cn²

T(1) = d

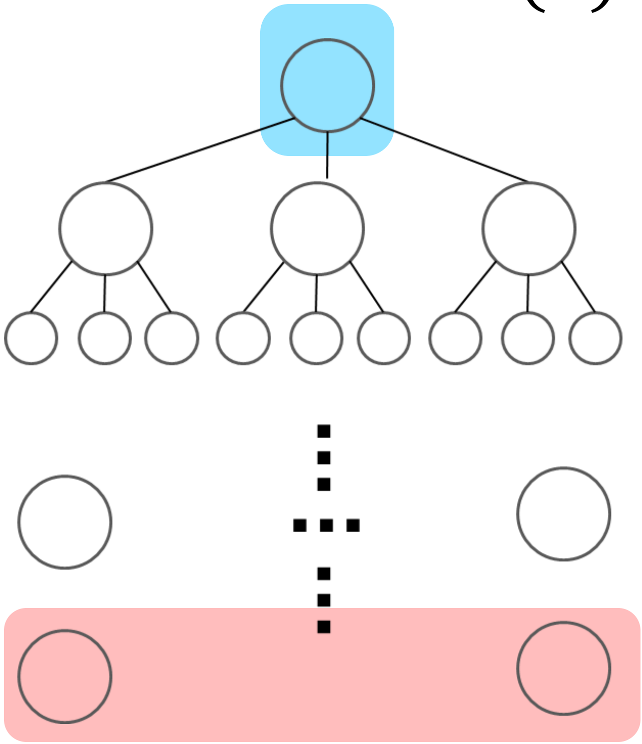
$$T(n) = dn^{\log_4 3} + cn^2 + \sum_{j=1}^{\log_4 n - 1} 3^j c\left(n/4^j\right)^2$$



T(n) = 3T(n/4) + cn^2

T(1) = d

$$T(n) = dn^{\log_4 3} + cn^2 + \sum_{j=1}^{\log_4 n - 1} 3^j c\left(n/4^j\right)^2$$

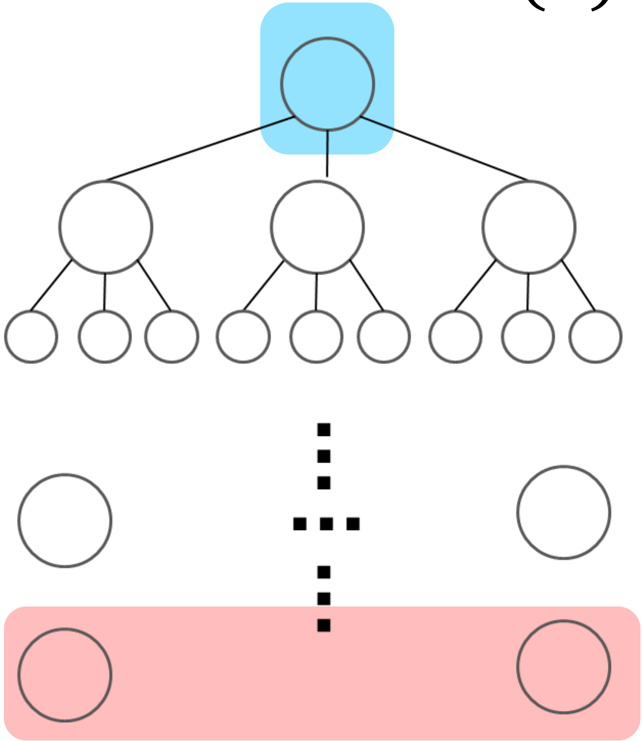


This term is from the *top-level call* (i.e. the root of the tree).

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

$T(1) = d$

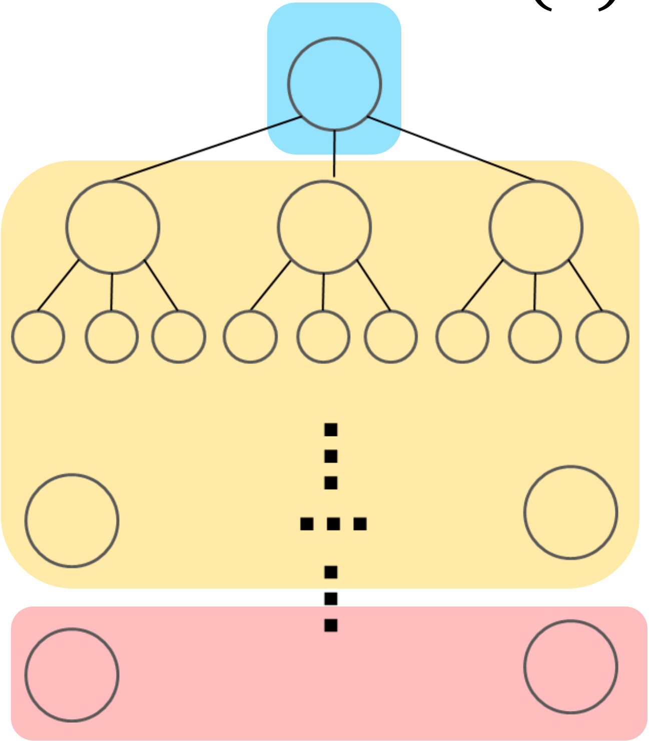
$$T(n) = dn^{\log_4 3} + cn^2 + \sum_{j=1}^{\log_4 n - 1} 3^j c\left(n/4^j\right)^2$$



T(n) = 3T(n/4) + cn²

T(1) = d

$$T(n) = dn^{\log_4 3} + cn^2 + \sum_{j=1}^{\log_4 n-1} 3^j c\left(n/4^j\right)^2$$



This term is from the *non-base-case recursive calls* (i.e. the rest of the tree).

Let’s Generalize

- We split up the sum for a particular recurrence

$$T(n) = 3T(n/4) + cn^2; \; T(1) = d$$

Let’s Generalize

- We split up the sum for a particular recurrence

$$T(n) = 3T(n/4) + cn^2; T(1) = d$$

- Let’s do this for a **general recurrence**

$$T(n) = aT(n/b) + f(n); T(1) = d$$

Let’s Generalize

- We split up the sum for a particula

$$T(n) = 3T(n/4) + cn^2;$$

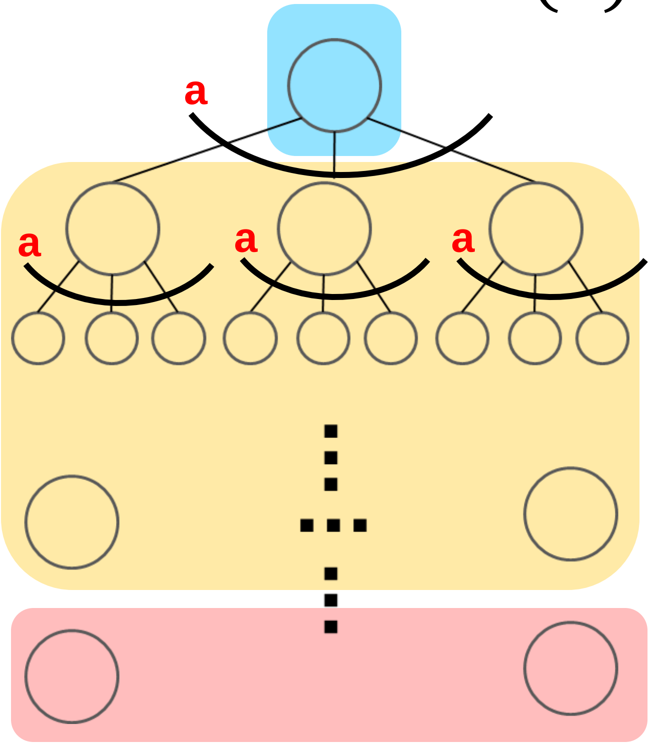
- Let’s do this for a **general recurre**

$$T(n) = aT(n/b) + f(n); \boxed{T(1) = d}$$

As you saw in Studio 4, we could start from $T(c_0)$ rather than $T(1)$; would not affect **asymptotic** result.

T(n) = aT(n/b) + f(n)
T(1) = d

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

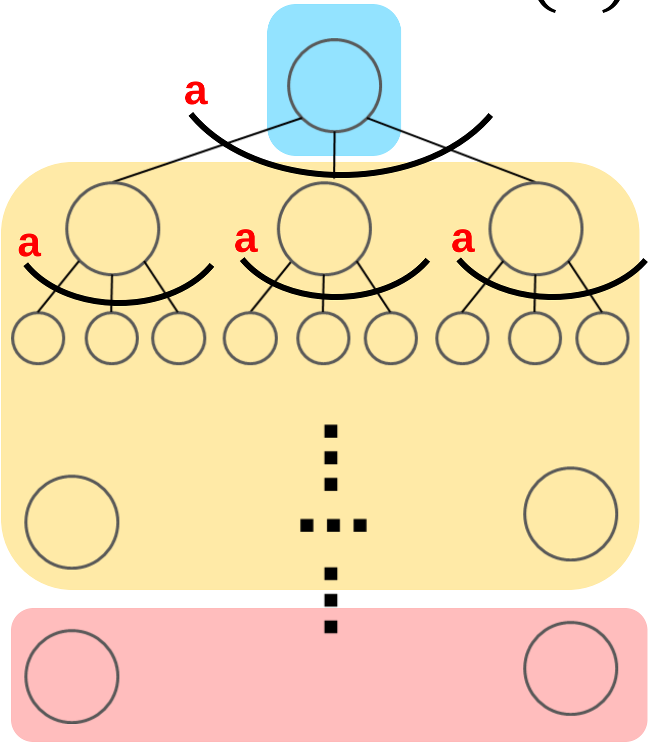


Which term, if any, dominates the sum?

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

T(1) = d

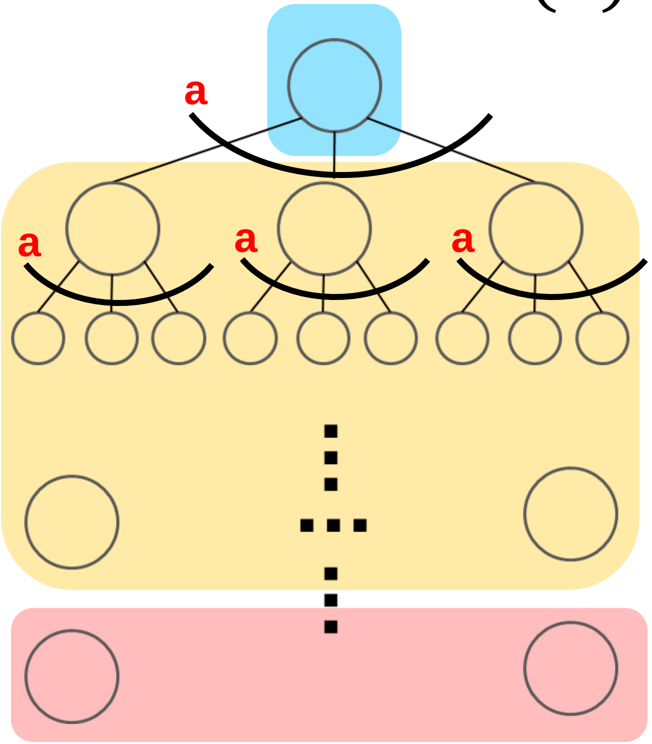
$$T(n) = dn^{\log_b a} + \boldsymbol{f(n)} + \sum_{j=1}^{\log_b n - 1} a^j f(n/b^j)$$



If top-of-tree work dominates,
T(n) = ???

T(n) = aT(n/b) + f(n)
T(1) = d

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

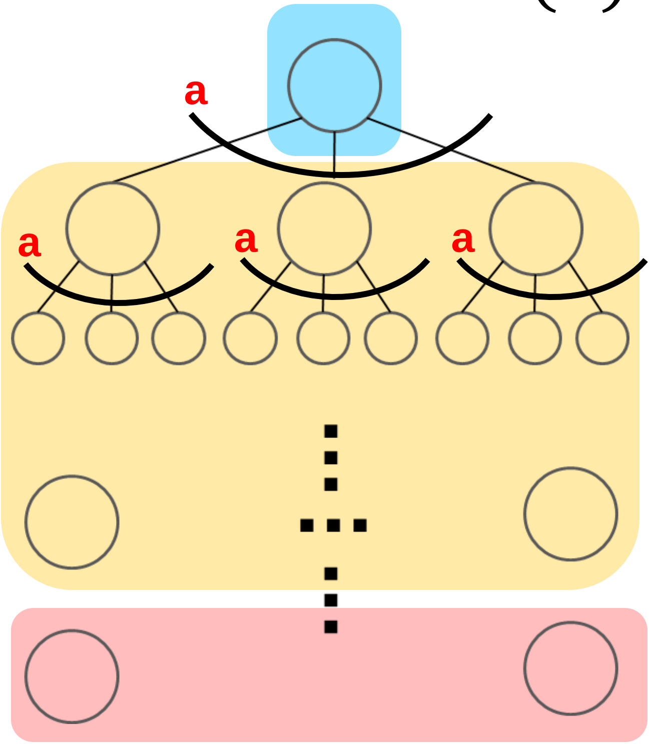


**If top-of-tree work dominates,
T(n) = Θ(f(n))**

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

$T(1) = d$

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

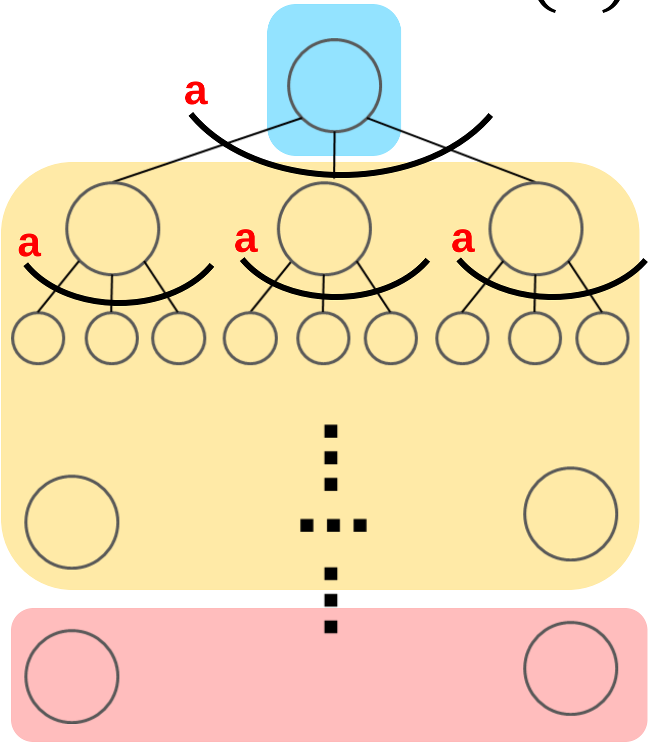


If bottom-of-tree work dominates,
$T(n) =$???

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

T(1) = d

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



If bottom-of-tree work dominates,
$T(n) = \Theta(n^{\log_b a})$

What if the top and bottom work *balance*?

What does “balance” mean?

- Top and bottom work are asymptotically the same.
- In other words,

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

What does “balance” mean?

- Top and bottom work are asymptotically the same.
- In other words,

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

For intuition, we’ll pretend that $f(n) = cn^{\log_b a}$

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

T(1) = d

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

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

T(1) = d

$$T(n) = dn^{\log_b a} + cn^{\log_b a} + \sum_{j=1}^{\log_b n - 1} a^j c\left(\frac{n}{b^j}\right)^{\log_b a}$$

T(n) = aT(n/b) + f(n)
T(1) = d

$$T(n) = dn^{\log_b a} + \sum_{j=0}^{\log_b n - 1} a^j c\left(\frac{n}{b^j}\right)^{\log_b a}$$

T(n) = aT(n/b) + f(n)
T(1) = d

$$T(n) = dn^{\log_b a} + cn^{\log_b a} \sum_{j=0}^{\log_b n - 1} a^j \left(\frac{1}{b^j}\right)^{\log_b a}$$

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

T(1) = d

$$T(n) = dn^{\log_b a} + cn^{\log_b a} \sum_{j=0}^{\log_b n - 1} \frac{a^j}{\left(b^{\log_b a}\right)^j}$$

T(n) = aT(n/b) + f(n)
T(1) = d

$$T(n) = dn^{\log_b a} + cn^{\log_b a} \sum_{j=0}^{\log_b n - 1} \frac{a^j}{a^j}$$

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

T(1) = d

$$T(n) = dn^{\log_b a} + cn^{\log_b a} \sum_{j=0}^{\log_b n-1} 1$$

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

T(1) = d

$$T(n) = dn^{\log_b a} + cn^{\log_b a} \log_b n$$

T(n) = aT(n/b) + f(n)
T(1) = d

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

When top and bottom of tree balance, all levels contribute equally to sum – and there are Θ(log n) levels.

Summary of Intuition

- Given recurrence T(n) = aT(n/b) + f(n)…
- If f(n) dominates $n^{\log_b a}$, then solution should be Θ(f(n))
- If $n^{\log_b a}$ dominates f(n), then solution should be Θ($n^{\log_b a}$)
- If f(n) = Θ($n^{\log_b a}$) [balance], then solution should be Θ(f(n) log n)

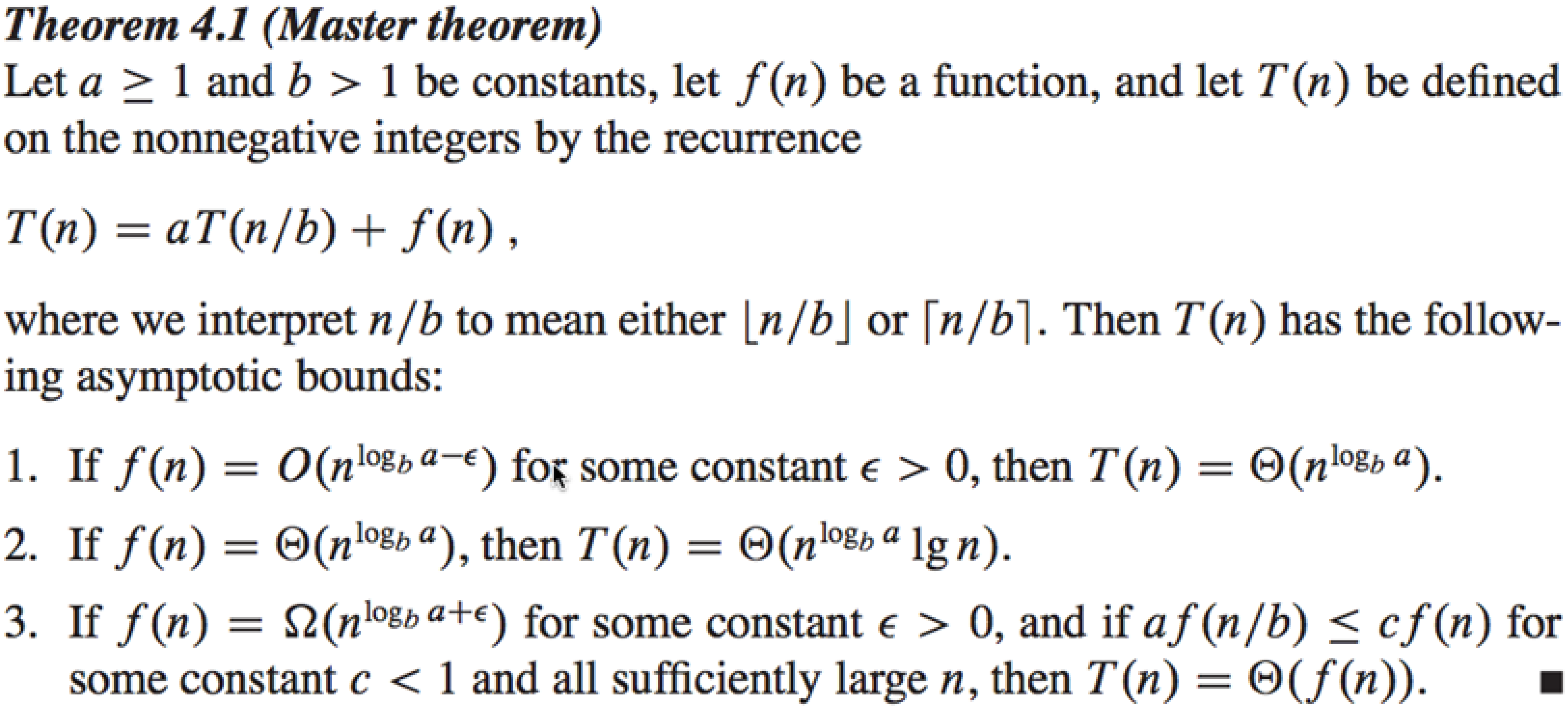
Summary of Intuition

- Given recurrence T(n) = aT(n/b)
- If f(n) dominates $n^{\log_b a}$, then so
- If $n^{\log_b a}$ dominates f(n), then so
- If f(n) = Θ($n^{\log_b a}$) [balance], the

This is not yet a theorem – in part because we haven’t carefully defined “dominates,” and in part because we didn’t do a careful proof.

So is there a theorem that captures our intuition?

Master Theorem (p. 94 of text)



Theorem 4.1 (Master theorem)

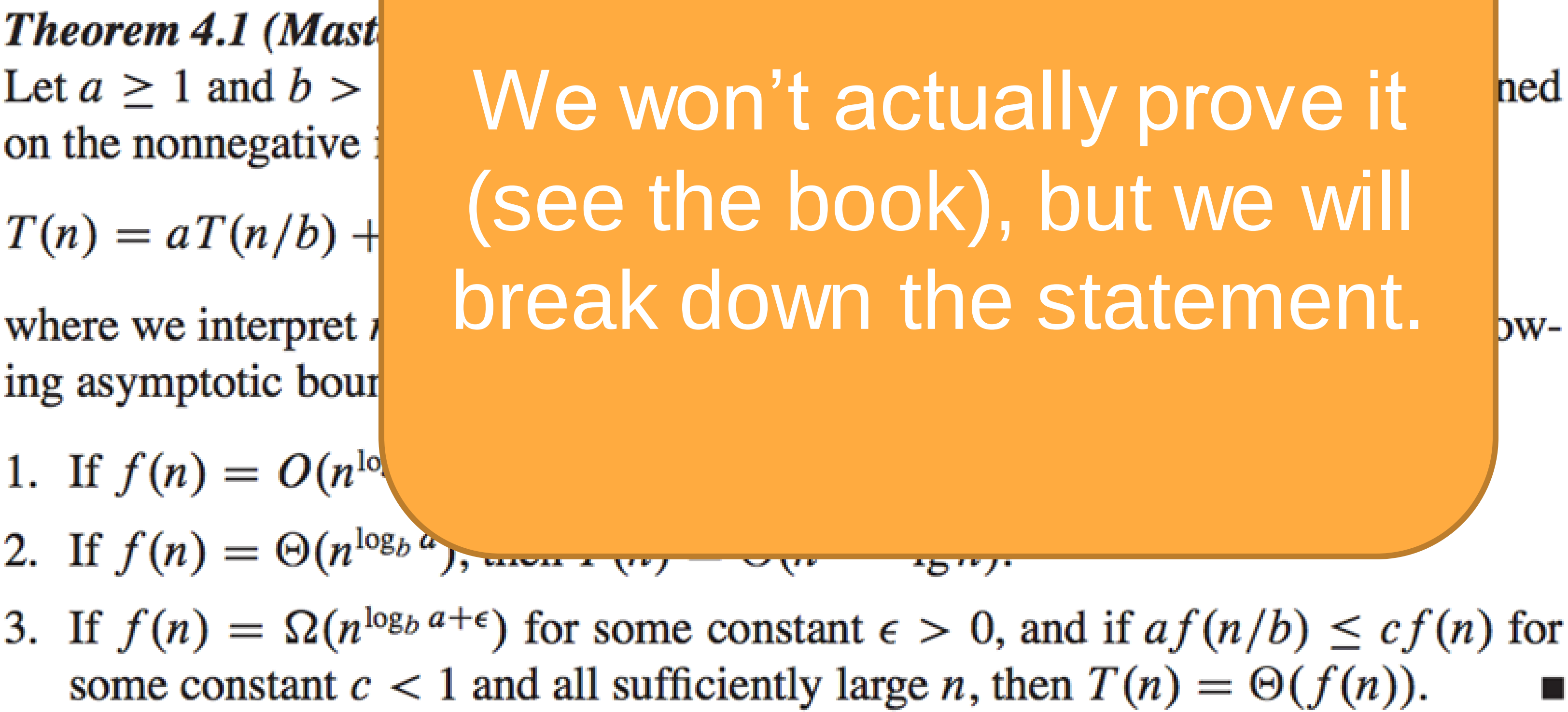
Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence

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

where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ has the following asymptotic bounds:

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \lg n)$.
3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$. ■

Master The



Theorem 4.1 (Mast

Let $a \ge 1$ and $b >$ ned

on the nonnegative i

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

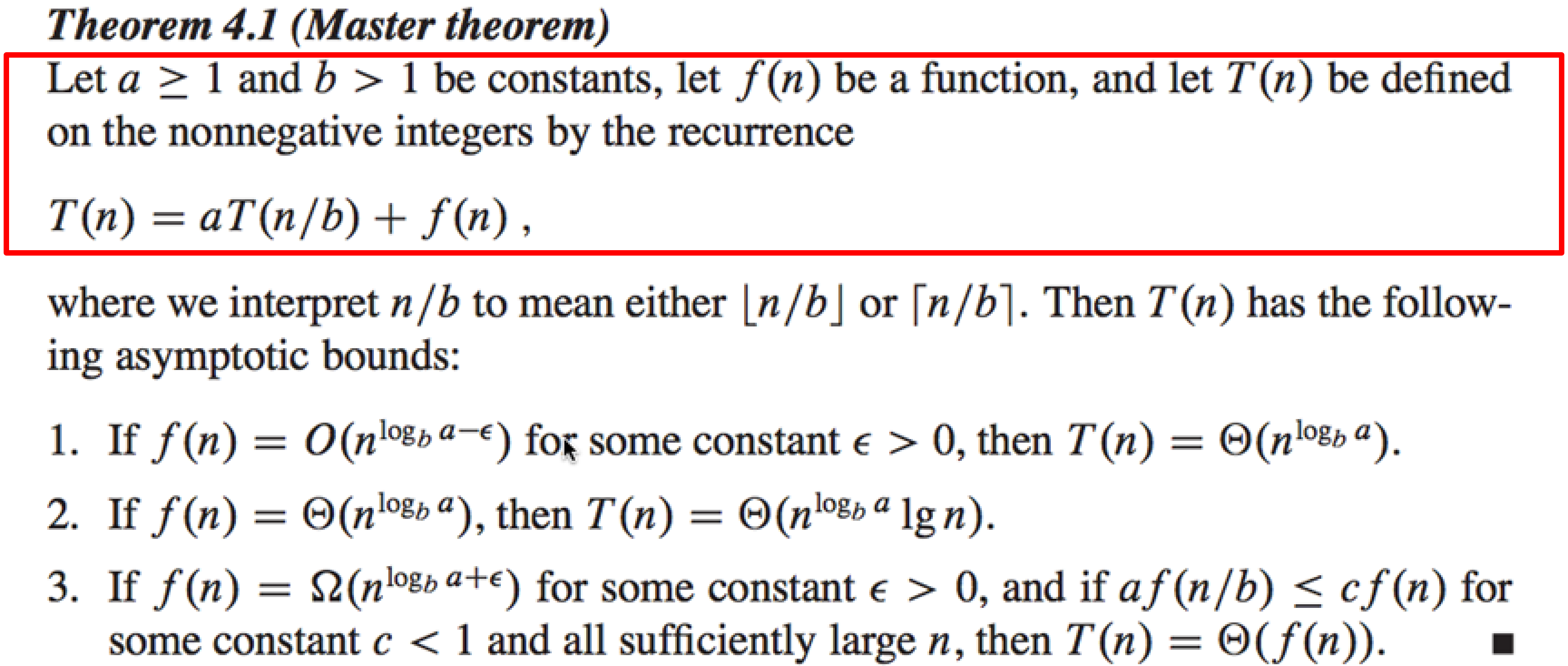
where we interpret n ow-

ing asymptotic bour

1. If $f(n) = O(n^{\mathrm{lo}}$
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \lg n)$.
3. If $f(n) = \Omega(n^{\log_b a+\epsilon})$ for some constant $\epsilon > 0$, and if $af(n/b) \le cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$. ■

We won’t actually prove it (see the book), but we will break down the statement.

Master Theorem (p. 94 of text)



Theorem 4.1 (Master theorem)

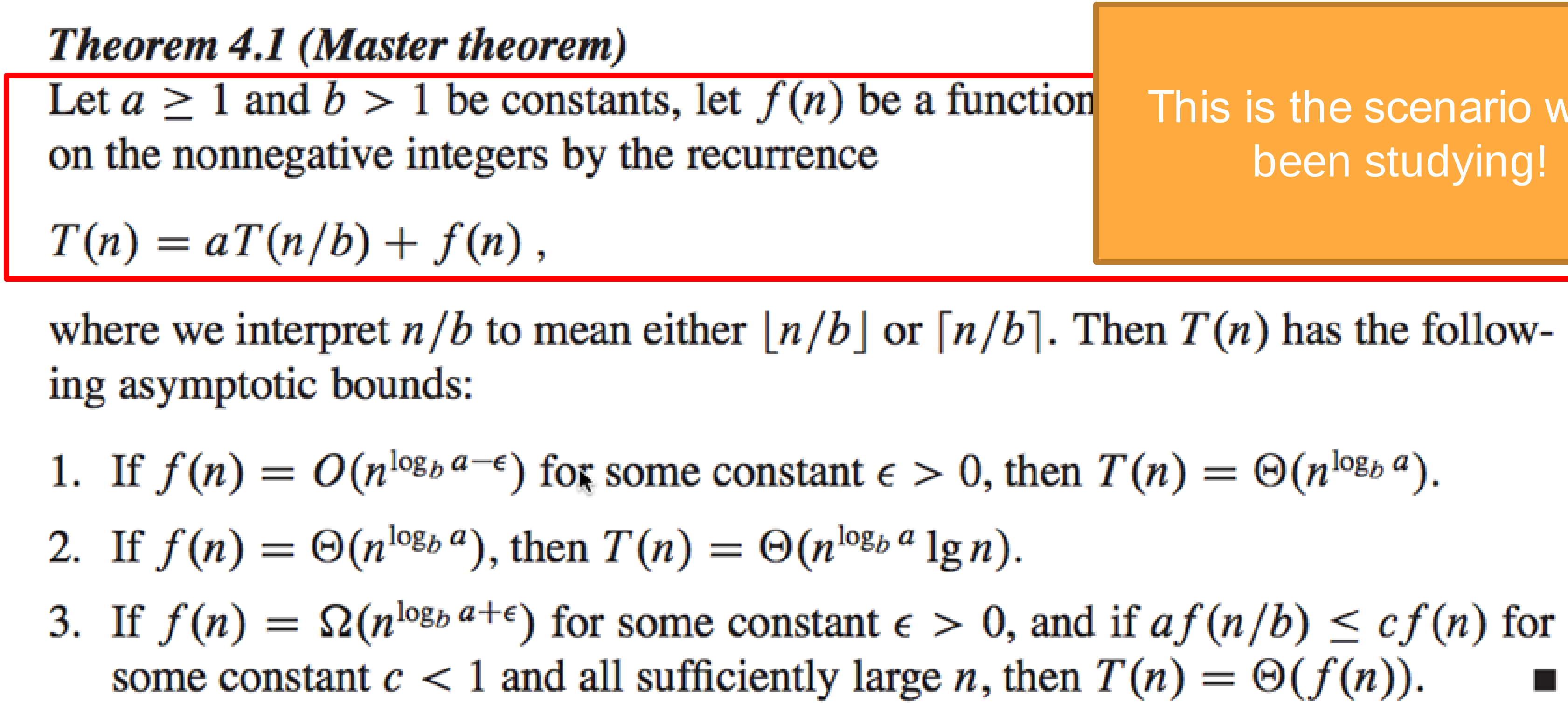
Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence

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

where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ has the following asymptotic bounds:

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \lg n)$.
3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$. ■

Master Theorem (p. 94 of text)



Theorem 4.1 (Master theorem)

Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function on the nonnegative integers by the recurrence

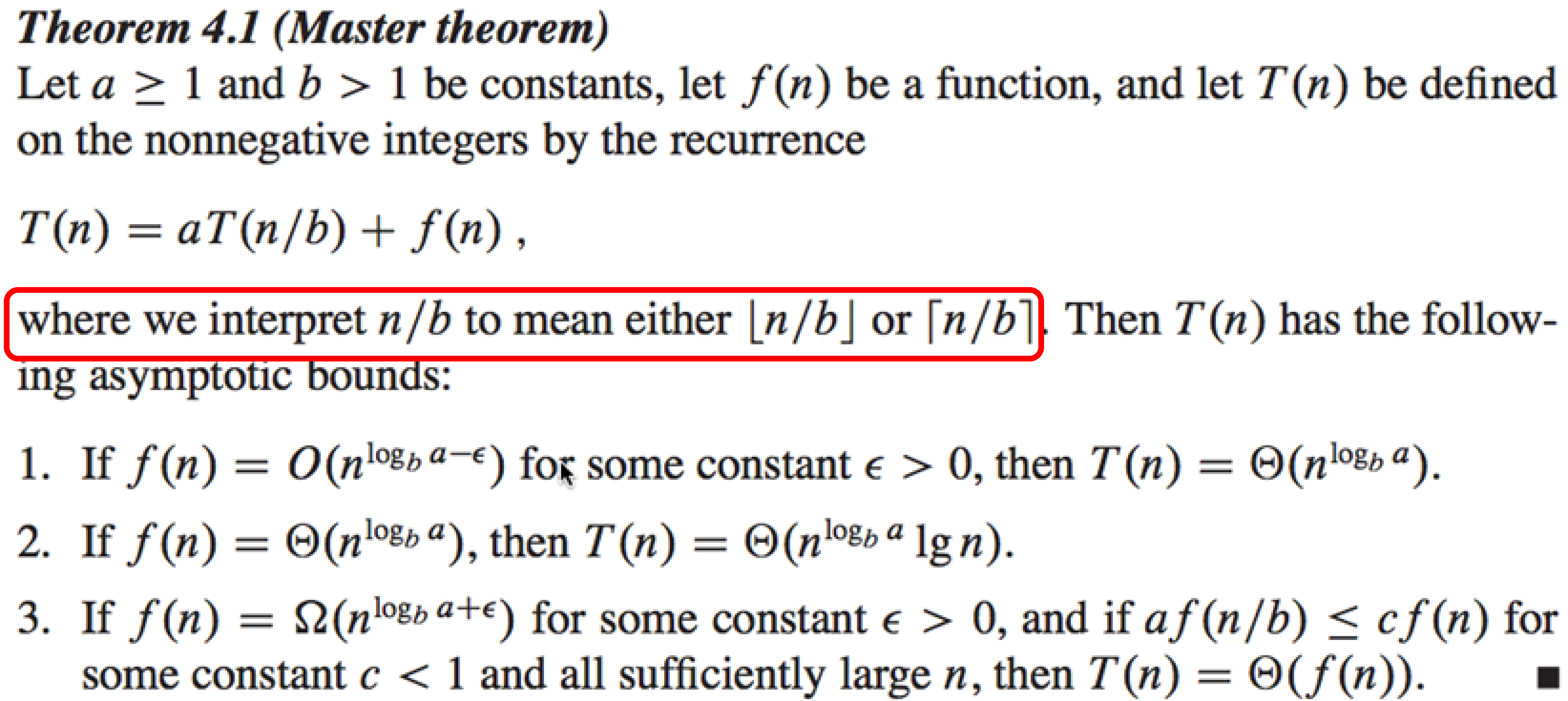
$$T(n) = aT(n/b) + f(n),$$

where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ has the following asymptotic bounds:

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \lg n)$.
3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$. ■

This is the scenario we've been studying!

Master Theorem (p. 94 of text)



Theorem 4.1 (Master theorem)

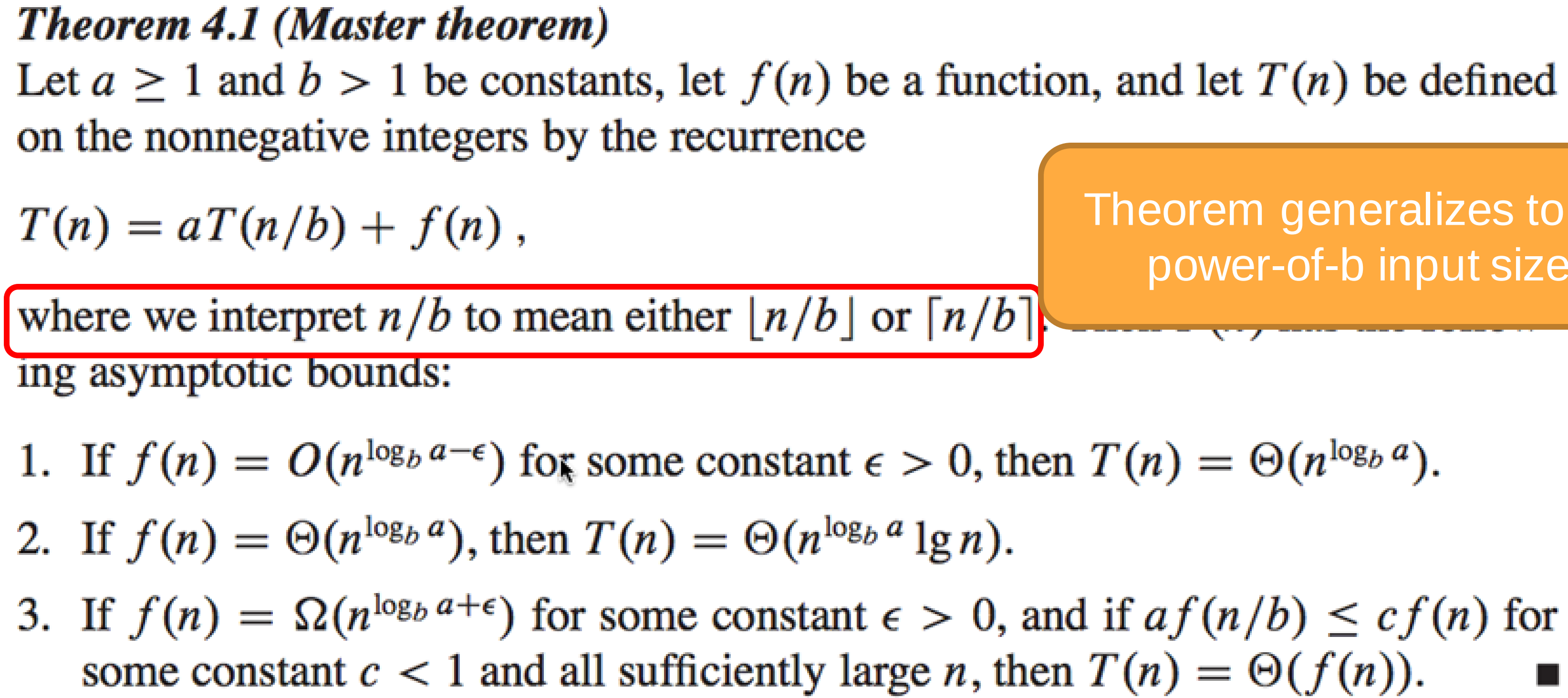
Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence

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

where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ has the following asymptotic bounds:

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \lg n)$.
3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$. ■

Master Theorem (p. 94 of text)



Theorem 4.1 (Master theorem)

Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence

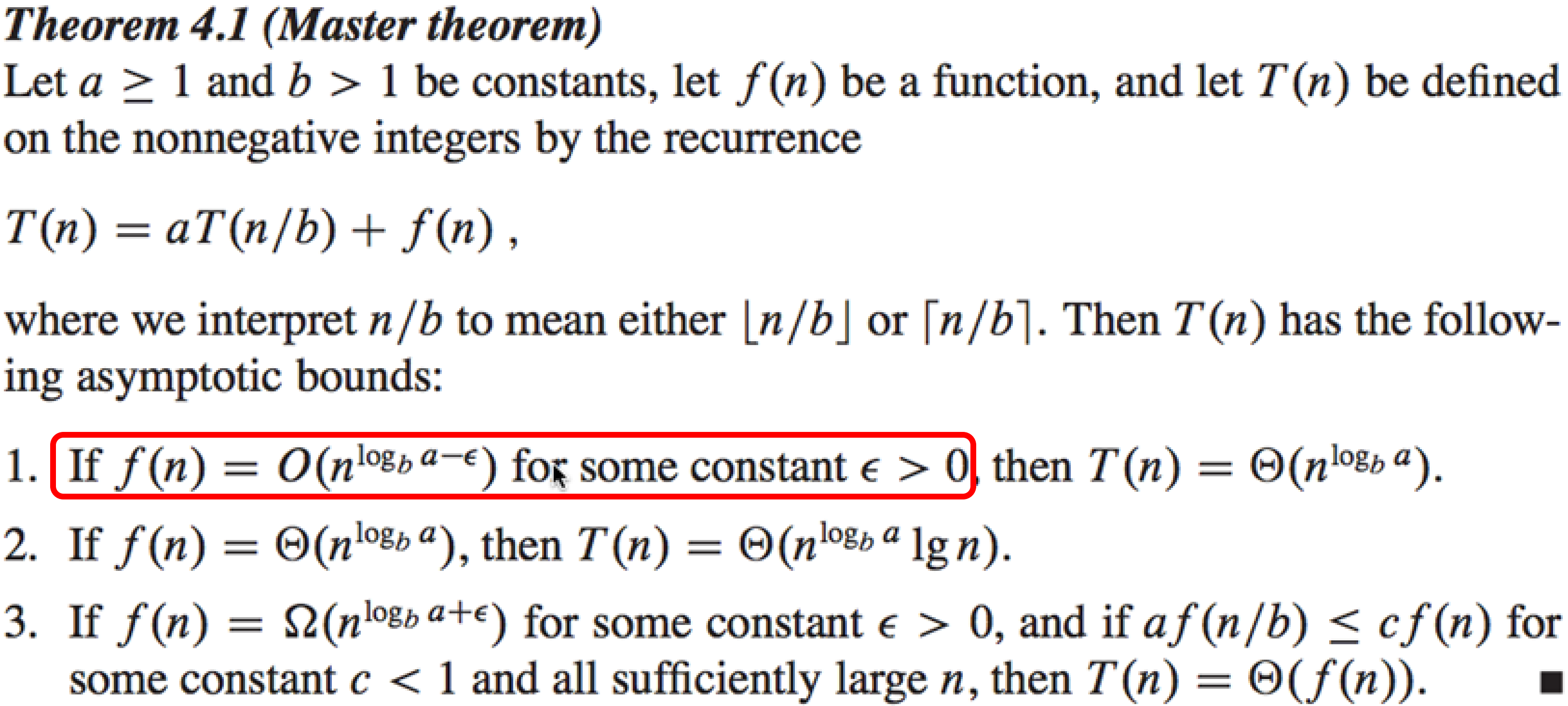
$$T(n) = aT(n/b) + f(n) ,$$

where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$.
ing asymptotic bounds:

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \lg n)$.
3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$. ■

Theorem generalizes to non-power-of-b input sizes!

Master Theorem (p. 94 of text)



Theorem 4.1 (Master theorem)

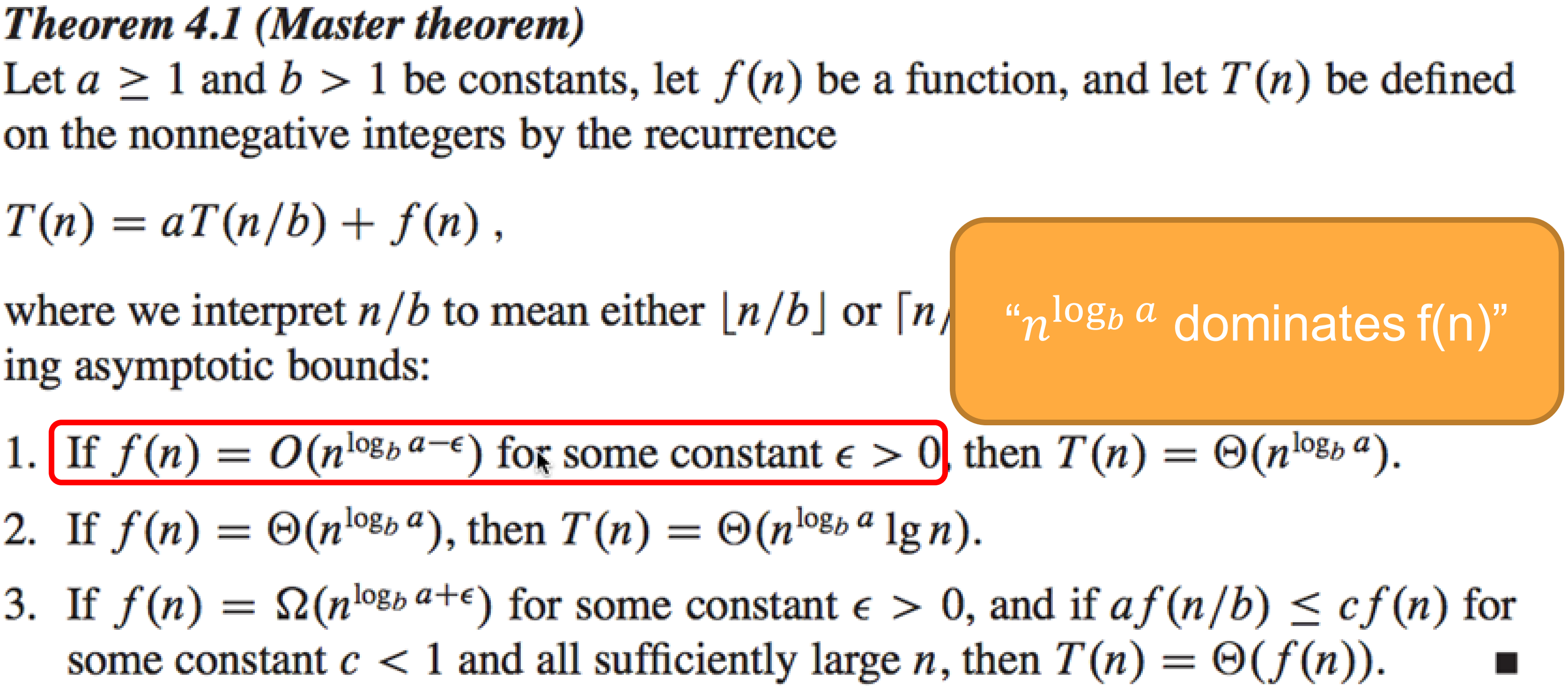
Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence

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

where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ has the following asymptotic bounds:

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \lg n)$.
3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$. ■

Master Theorem (p. 94 of text)



Theorem 4.1 (Master theorem)

Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence

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

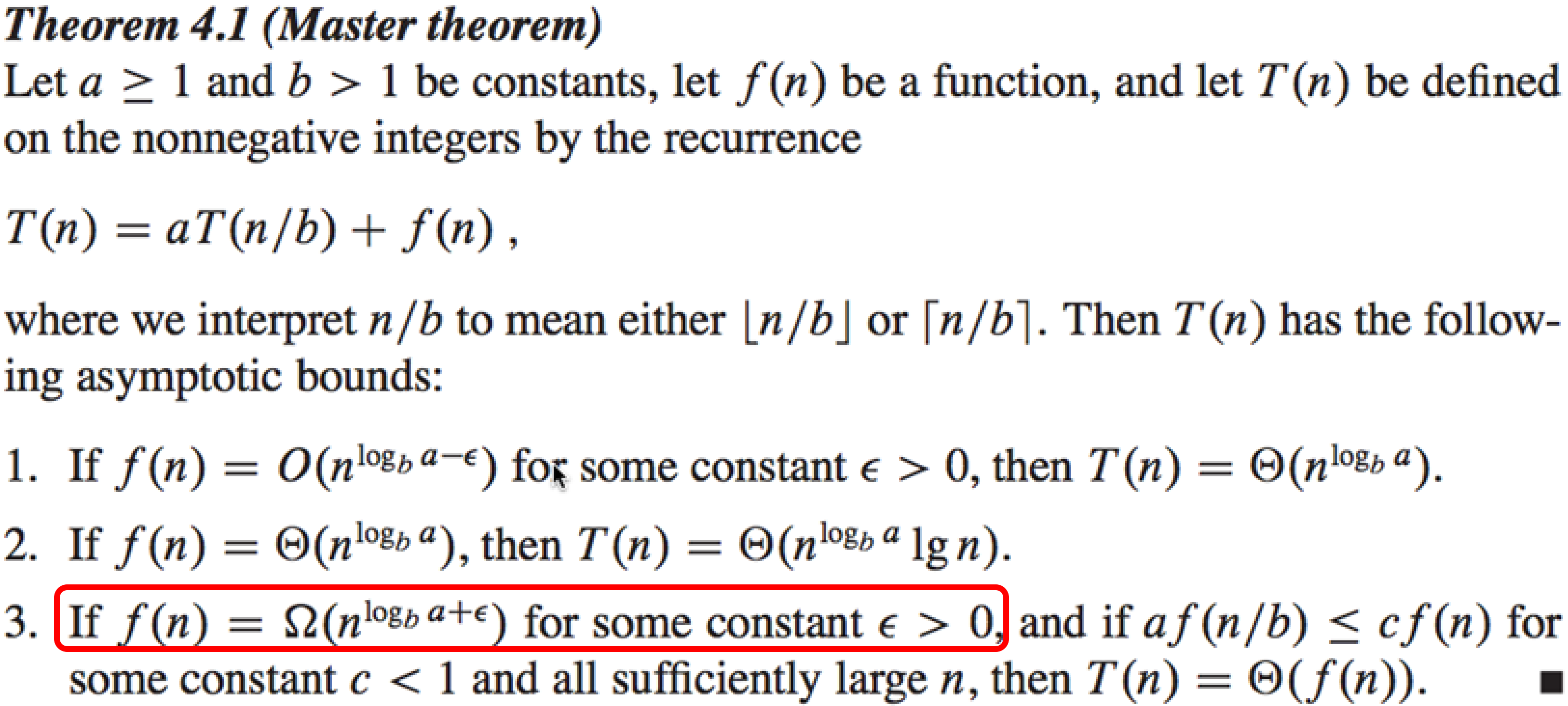
where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/$

ing asymptotic bounds:

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \lg n)$.
3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$. ■

"$n^{\log_b a}$ dominates f(n)"

Master Theorem (p. 94 of text)



Theorem 4.1 (Master theorem)

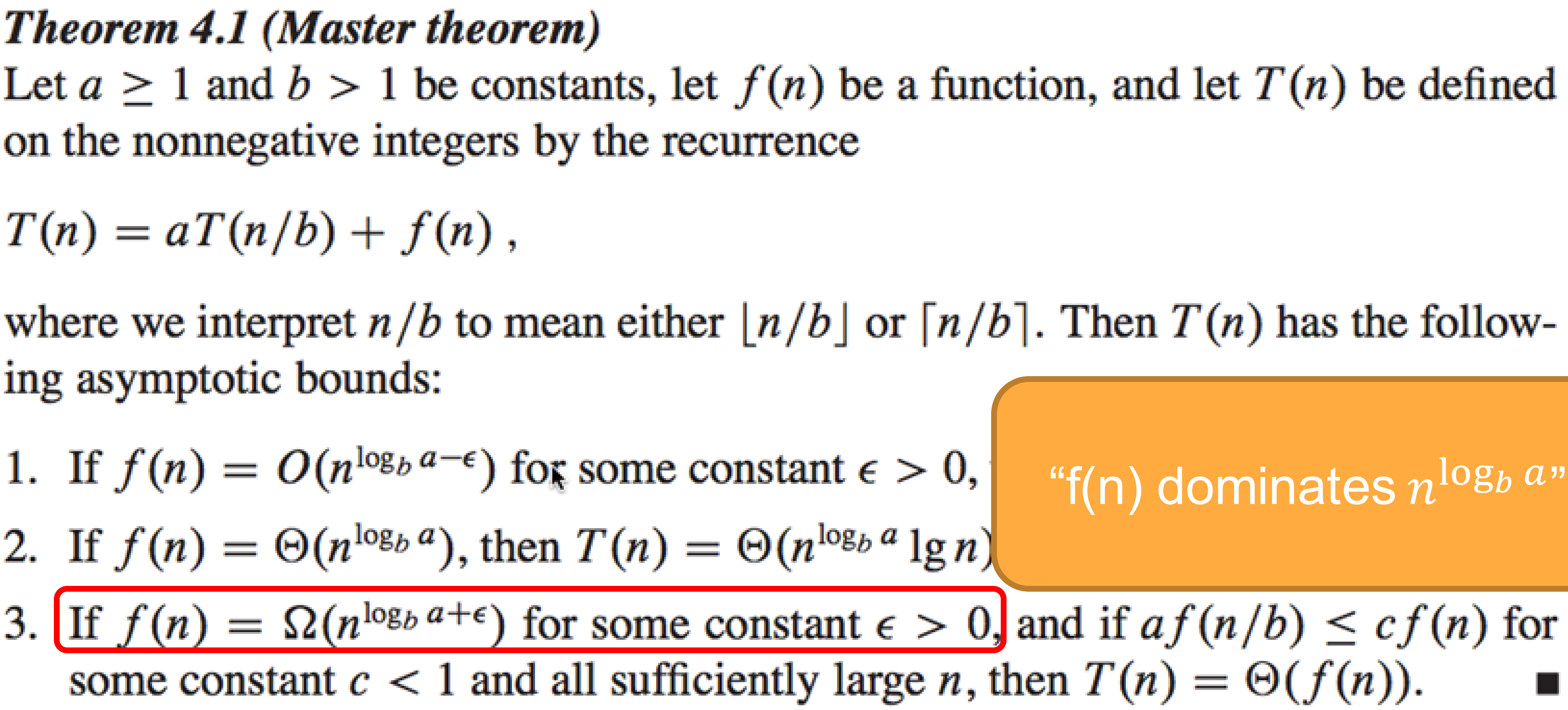
Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence

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

where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ has the following asymptotic bounds:

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \lg n)$.
3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$. ■

Master Theorem (p. 94 of text)



Theorem 4.1 (Master theorem)

Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence

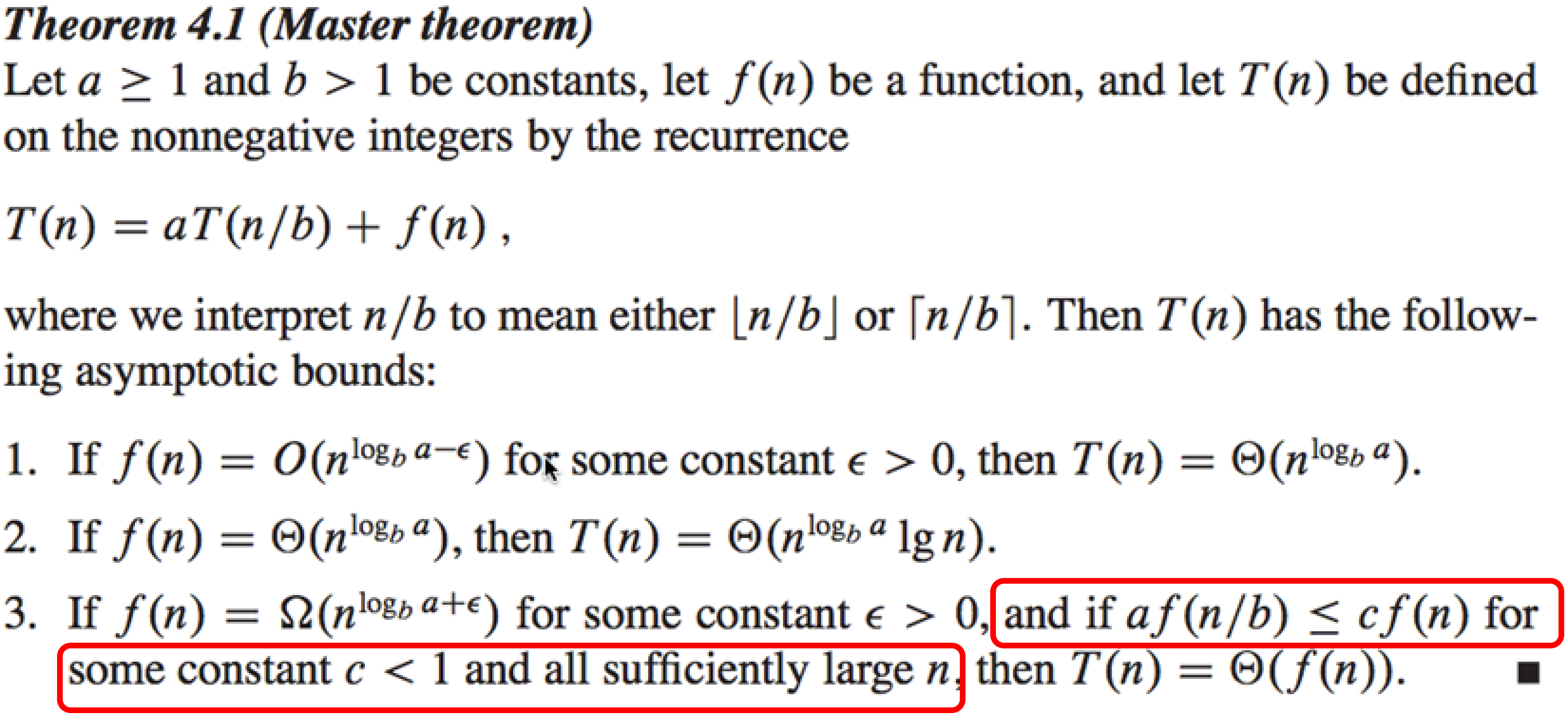
$$T(n) = aT(n/b) + f(n) ,$$

where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ has the following asymptotic bounds:

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$,
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \lg n)$
3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$. ■

“f(n) dominates $n^{\log_b a}$”

Master Theorem (p. 94 of text)



Theorem 4.1 (Master theorem)

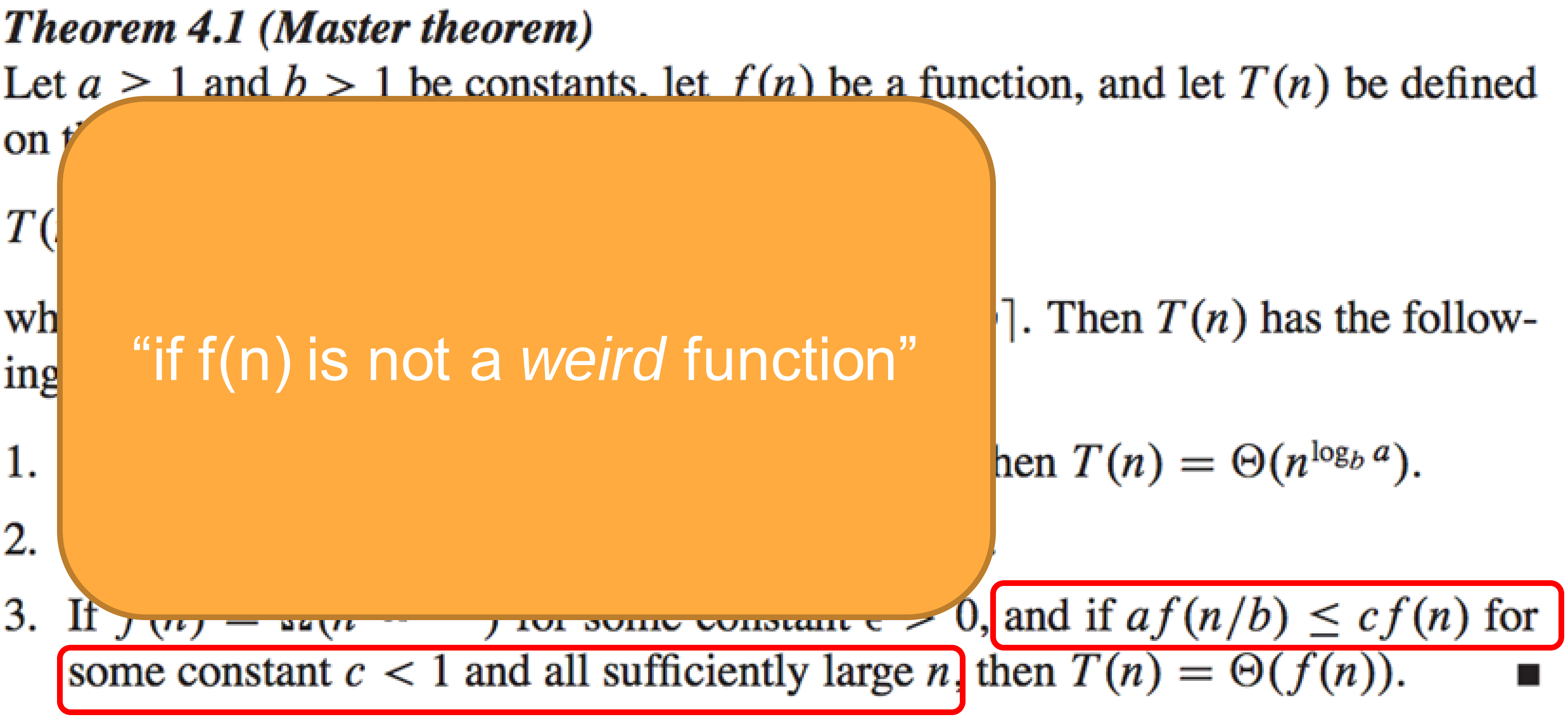
Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence

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

where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ has the following asymptotic bounds:

1. If $f(n) = O(n^{\log_b a-\epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \lg n)$.
3. If $f(n) = \Omega(n^{\log_b a+\epsilon})$ for some constant $\epsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$. ■

Master Theorem (p. 94 of text)



Theorem 4.1 (Master theorem)

Let $a > 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on t

$T($

wh]. Then $T(n)$ has the following

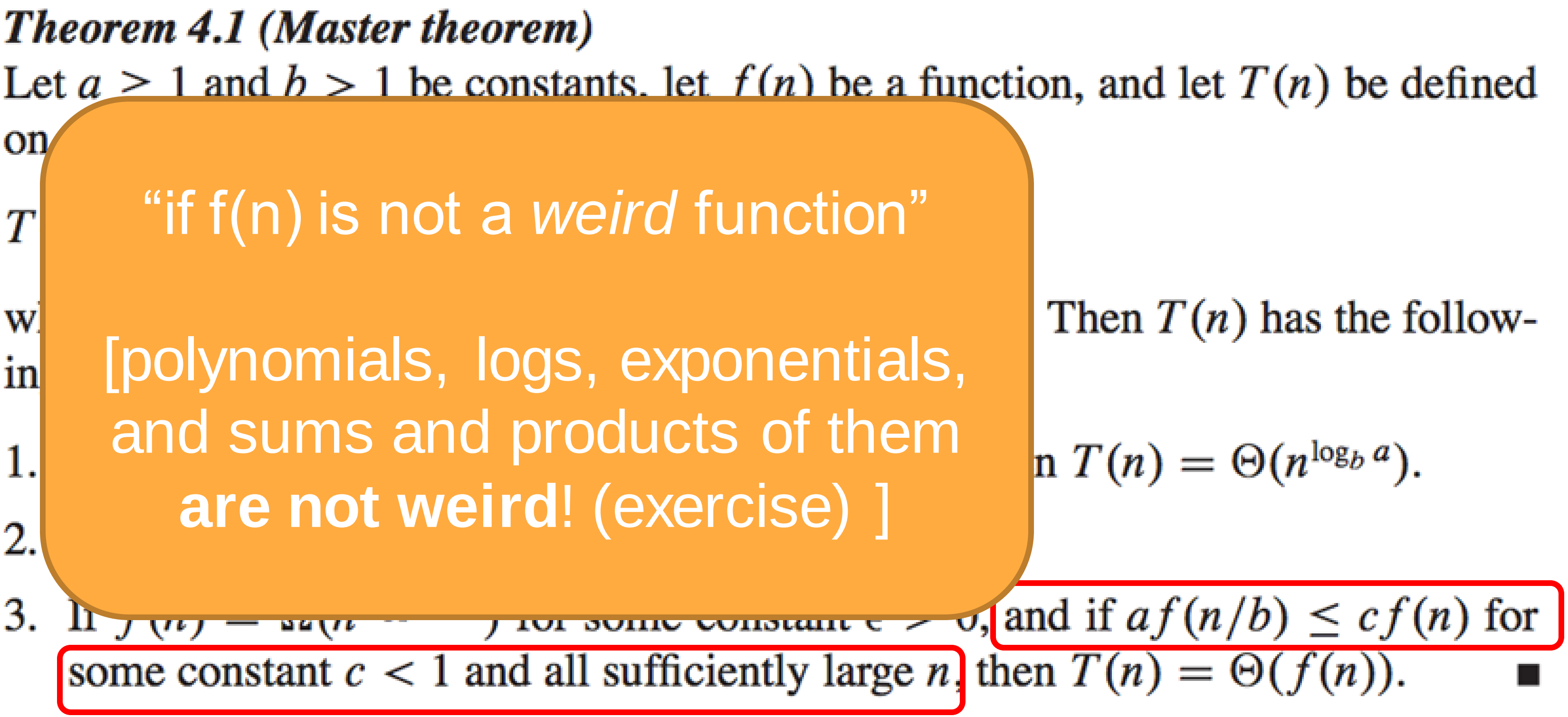
1. hen $T(n) = \Theta(n^{\log_b a})$.

2.

3. If 0, and if $af(n/b) \le cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$. ■

“if f(n) is not a *weird* function”

Master Theorem (p. 94 of text)



Theorem 4.1 (Master theorem)

Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined

on

T

w Then $T(n)$ has the follow-

in

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

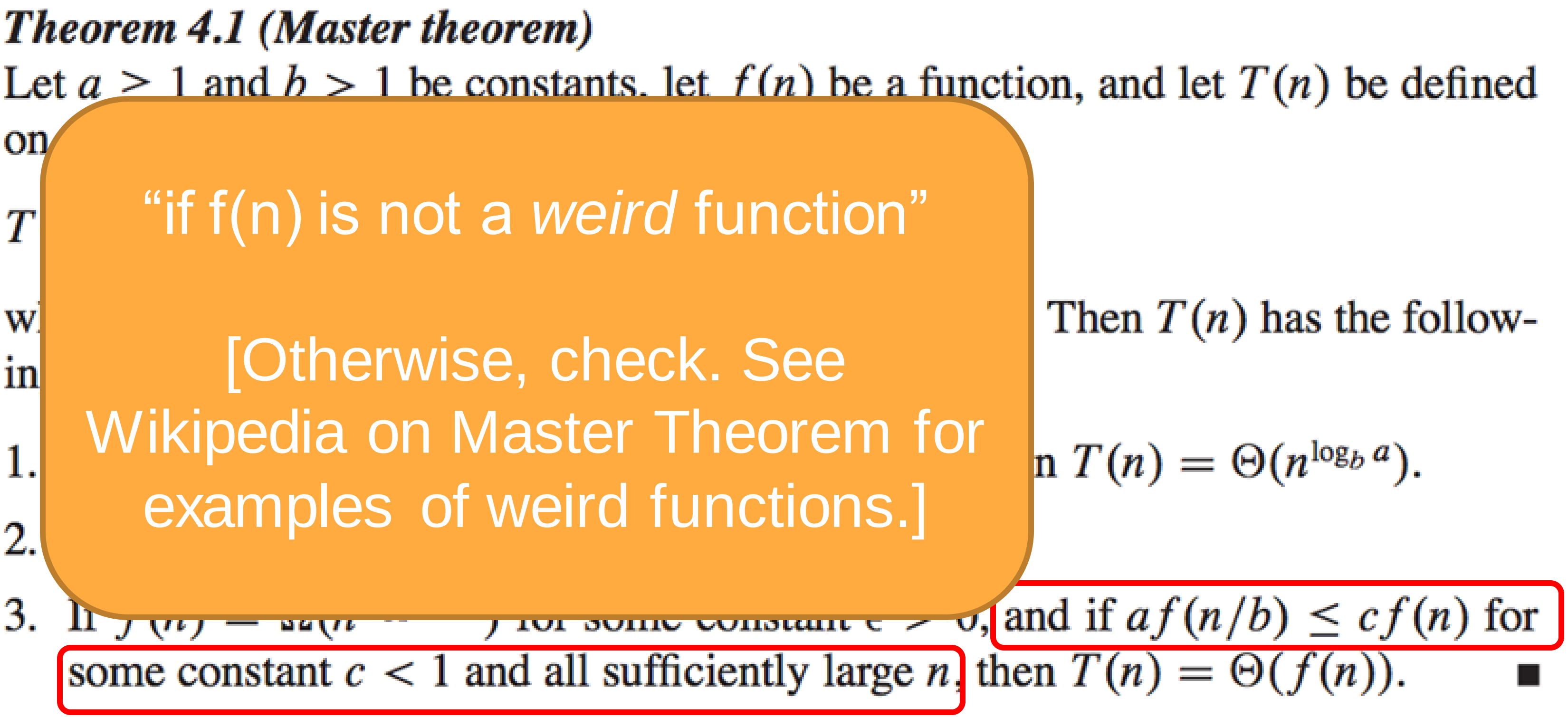
2.

3. If $f(n) = \Omega(n$ $)$ for some constant $\epsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$. ■

"if f(n) is not a *weird* function"

[polynomials, logs, exponentials, and sums and products of them **are not weird**! (exercise)]

Master Theorem (p. 94 of text)



Theorem 4.1 (Master theorem)

Let $a \ge 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined

on

T

w Then $T(n)$ has the follow-

in

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

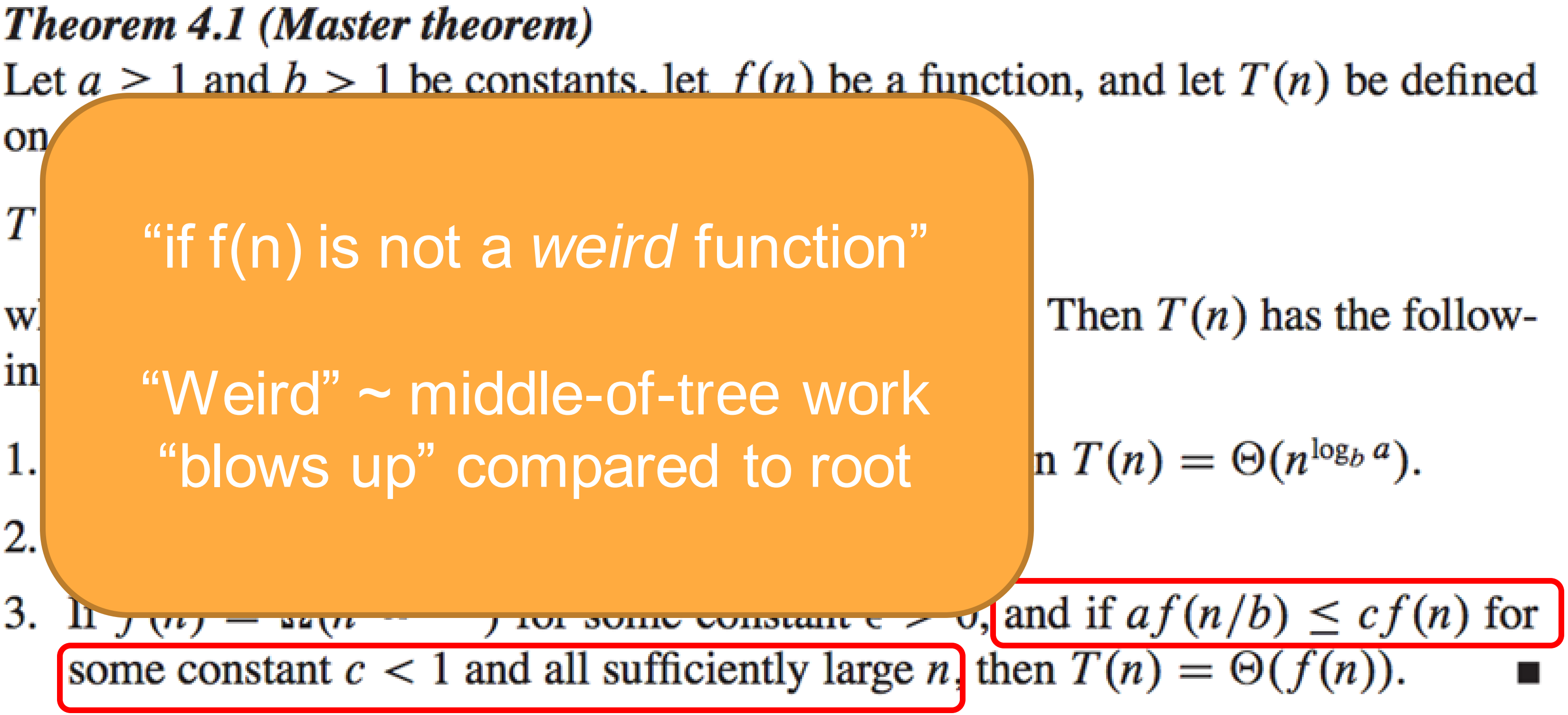
2.

3. If $f(n) = \Omega(n \;\;)$ for some constant $\epsilon > 0$, and if $af(n/b) \le cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$. ■

“if f(n) is not a *weird* function”

[Otherwise, check. See Wikipedia on Master Theorem for examples of weird functions.]

Master Theorem (p. 94 of text)



Theorem 4.1 (Master theorem)

Let $a > 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined

on

T

w Then $T(n)$ has the follow-

in

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

2.

3. If $f(n) = \Omega(n$) for some constant $\epsilon > 0$, and if $af(n/b) \le cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$. ■

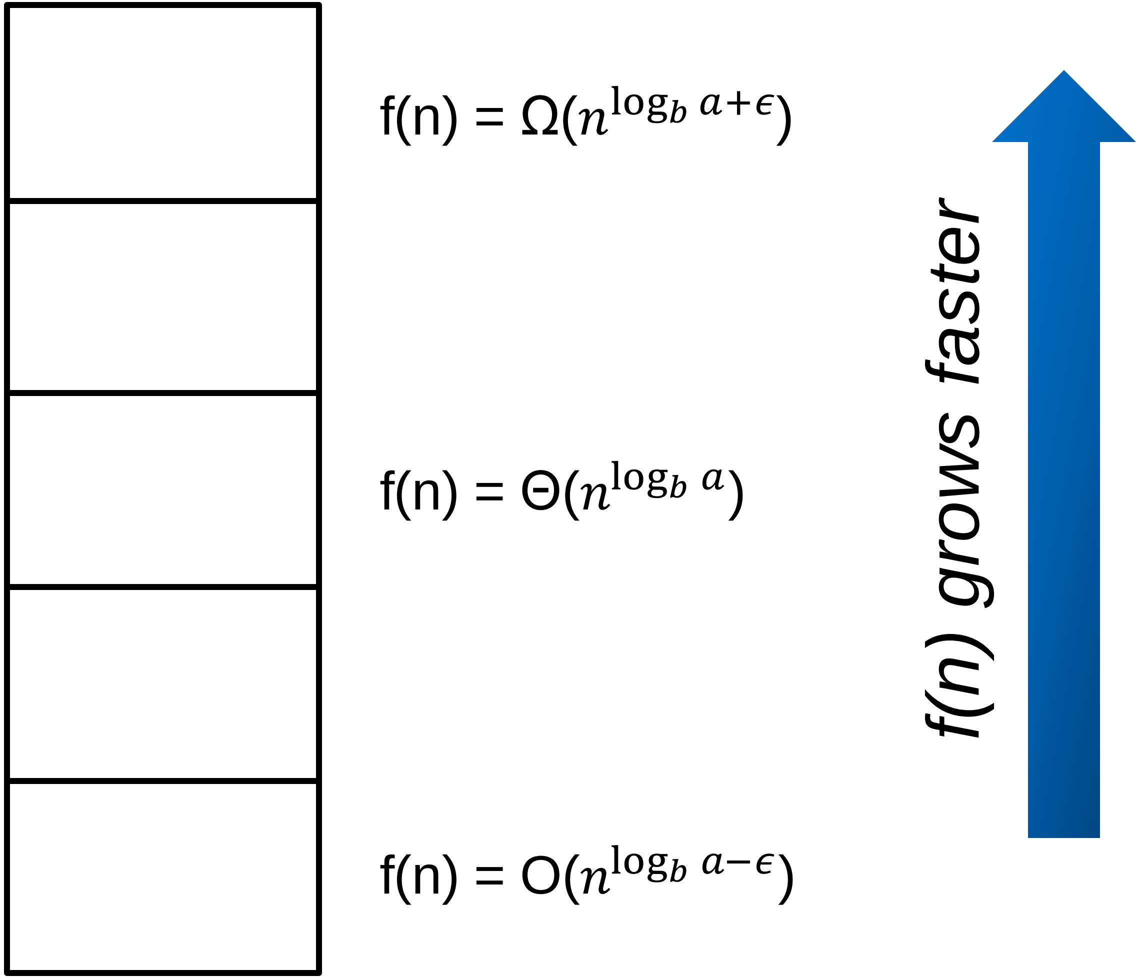
“if f(n) is not a *weird* function”

“Weird” ~ middle-of-tree work “blows up” compared to root

Key Elaboration of Theorem vs Intuition

- Precisely defines “**dominates**”
- “f(n) dominates g(n)” iff *f(n) grows* ***polynomially*** *faster than g(n)*
- *This is a* ***stronger condition*** *than* $f(n) = \omega(g(n))$

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



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

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

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

f(n) grows faster

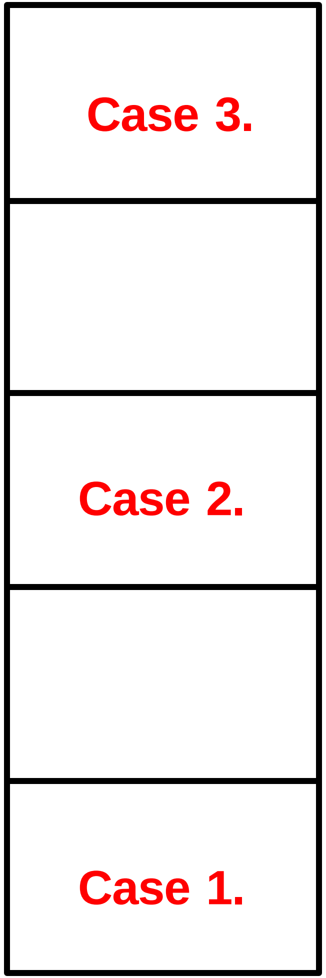
T(n) = a T(n/b) + f(n)

Case 3. f(n) = $\Omega(n^{\log_b a+\epsilon})$ → T(n) = Θ(f(n))

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

Case 1. f(n) = $O(n^{\log_b a-\epsilon})$ → T(n) = $\Theta(n^{\log_b a})$

T(n) = a T(n/b) + f(n)



Case 3.

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

f(n) = $\omega(n^{\log_b a})$, but f(n) = $o(n^{\log_b a+\epsilon})$

Case 2.

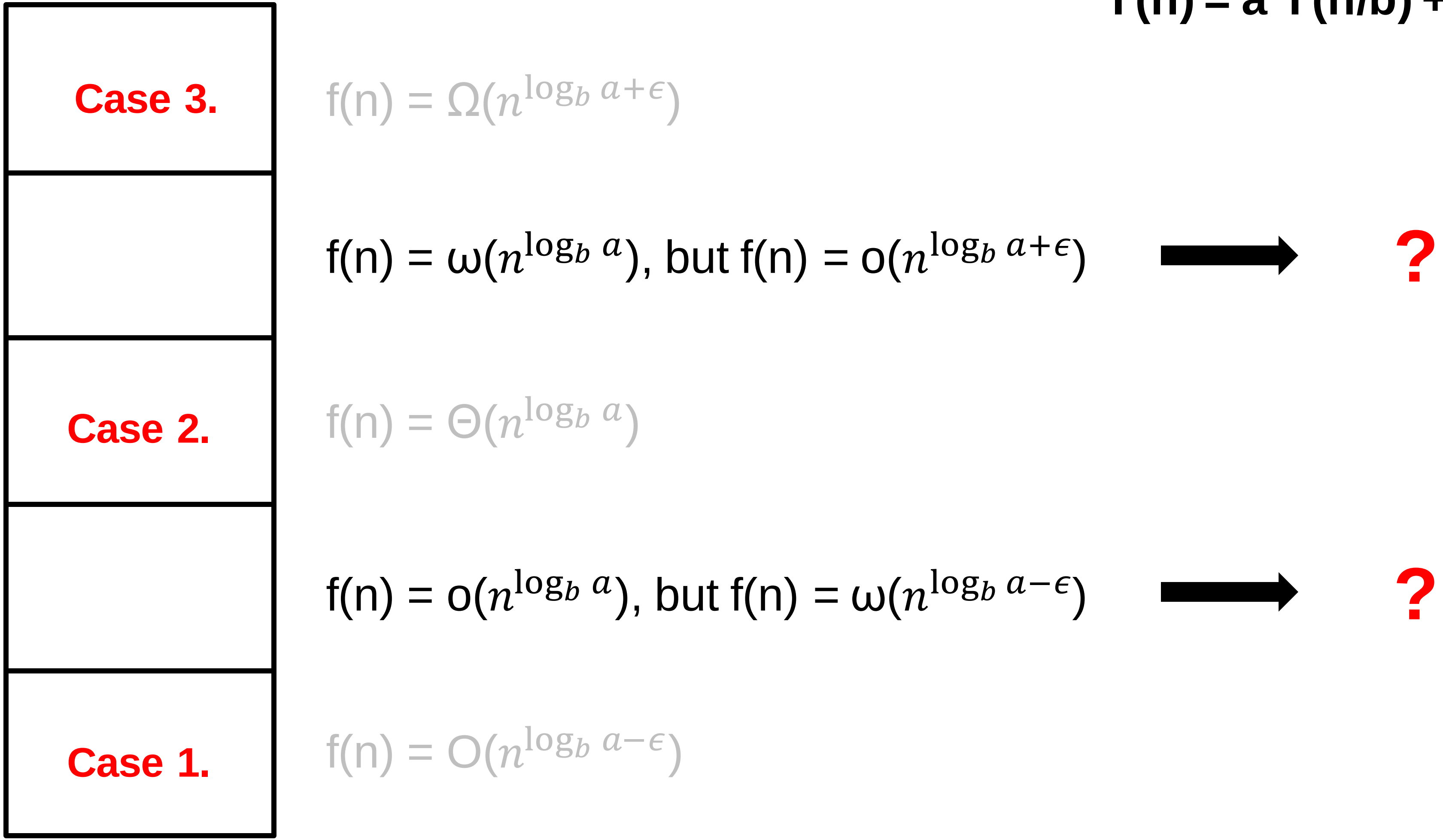
f(n) = $\Theta(n^{\log_b a})$

f(n) = $o(n^{\log_b a})$, but f(n) = $\omega(n^{\log_b a-\epsilon})$

Case 1.

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

T(n) = a T(n/b) + f(n)



Case 3. f(n) = $\Omega(n^{\log_b a+\epsilon})$

f(n) = $\omega(n^{\log_b a})$, but f(n) = $o(n^{\log_b a+\epsilon})$ → **?**

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

f(n) = $o(n^{\log_b a})$, but f(n) = $\omega(n^{\log_b a-\epsilon})$ → **?**

Case 1. f(n) = $O(n^{\log_b a-\epsilon})$

T(n) = a T(n/b) + f(n)

Case 3. f(n) = $\Omega(n^{\log_b a+\epsilon})$

f(n) = $\omega(n^{\log_b a})$, but f(n) = $o(n^{\log_b a+\epsilon})$ →

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

f(n) = $o(n^{\log_b a})$, but f(n) = $\omega(n^{\log_b a-\epsilon})$ →

Case 1. f(n) = $O(n^{\log_b a-\epsilon})$

Limits of the Master Theorem

- If the form of the recurrence does not match the statement of the theorem…
- …or the recurrence falls into “gap” between two cases…
- …then the Master Theorem does not apply.
- (You must find another way to solve the recurrence.)

Limits of the Master Theorem

- See the Wikipedia page on the Master Theorem

- Examples of situations where Master Thm doesn't apply
    - *Note:* a and b in Master Theorem don't *have* to be integers
        - (though for recursive programs, a is an integer – why?)
        - a must be ≥ 1
        - b must be > 1 – why?

- Example of function that fails non-weirdness condition

n log n vs. $n^{1+\varepsilon}$

- **Example recurrence: $T(n) = 2T(n/2) + n\log n$**
- Question: does Case 3 apply?
    - I.e. does n log n = $\Omega(n^{1+\varepsilon})$ for some $\varepsilon > 0$?

n log n vs. $n^{1+\varepsilon}$

- Example recurrence: $T(n) = 2T(n/2) + n\log n$
- Question: does Case 3 apply?
  - I.e. does $n \log n = \Omega(n^{1+\varepsilon})$ for some $\varepsilon > 0$?
- Analysis by limit test
  - $\lim\ (n \log n) / (n^{1+\varepsilon}) = \lim\ (\log n + 1) / (1+\varepsilon)n^{\varepsilon}$

n log n vs. $n^{1+\varepsilon}$

- Example recurrence: $T(n) = 2T(n/2) + n\log n$
- Question: does Case 3 apply?
  - I.e. does $n \log n = \Omega(n^{1+\varepsilon})$ for some $\varepsilon > 0$?
- Analysis by limit test
  - $\lim (n \log n) / (n^{1+\varepsilon}) = \lim (\log n + 1) / (1+\varepsilon)n^{\varepsilon}$
  - $= \lim (1/n) / \varepsilon(1+\varepsilon)n^{\varepsilon-1}$

n log n vs. $n^{1+\varepsilon}$

- Example recurrence: $T(n) = 2T(n/2) + n\log n$
- Question: does Case 3 apply?
  - I.e. does $n \log n = \Omega(n^{1+\varepsilon})$ for some $\varepsilon > 0$?
- Analysis by limit test
  - $\lim (n \log n) / (n^{1+\varepsilon}) = \lim (\log n + 1) / (1+\varepsilon)n^{\varepsilon}$
  - $= \lim (1/n) / \varepsilon(1+\varepsilon)n^{\varepsilon-1}$
  - $= \lim (1 / \varepsilon(1+\varepsilon)nn^{\varepsilon-1})$

n log n vs. $n^{1+\varepsilon}$

- Example recurrence: $T(n) = 2T(n/2) + n\log n$
- Question: does Case 3 apply?
  - I.e. does $n \log n = \Omega(n^{1+\varepsilon})$ for some $\varepsilon > 0$?
- Analysis by limit test
  - $\lim (n \log n) / (n^{1+\varepsilon}) = \lim (\log n + 1) / (1+\varepsilon)n^{\varepsilon}$
  - $= \lim (1/n) / \varepsilon(1+\varepsilon)n^{\varepsilon-1}$
  - $= \lim (1 / \varepsilon(1+\varepsilon)nn^{\varepsilon-1})$
  - $= \lim (1 / \varepsilon(1+\varepsilon)n^{\varepsilon}) = 0$, because $\varepsilon > 0$

- Hence, n log n is $o(n^{1+\varepsilon})$ for every $\varepsilon > 0$

- So **NO**, Case 3 of Master Theorem does not apply.

n log n vs. $n^{1+ε}$

- Example recurrence: T(n) = 2T(n/2) + nlogn
- Questio
    - I.e. c
- Analysi
    - lim (
    -
    -
    -

But for Studio 5, see Wiki for a more general “balanced case” that specifically allows for f(n) to have extra log terms.

- Hence, n

- So **NO**, Case 3 of Master Theorem does not apply.

A little practice with “polynomially larger”

	polynomially larger than?	
n^2		n
$n^2 \log n$		n^2
$n^3 \log n$		n^2
$n^{2.001}$		n^2
n log n		$n^{\log_4 3}$

A little practice with “polynomially larger”

	polynomially larger than?	
n^2	**YES**	n
n^2 log n	**NO**	n^2
n^3 log n	**YES**	n^2
$n^{2.001}$	**YES**	n^2
n log n	**???**	$n^{\log_4 3}$

A little practice with “polynomially larger”

	polynomially larger than?	
n^2	**YES**	n
$n^2 \log n$	**NO**	n^2
$n^3 \log n$	**YES**	n^2
$n^{2.001}$	**YES**	n^2
n log n	**YES!**	$n^{\log_4 3}$

Applying the Master Theorem

- $T(n) = 2T(n/2) + cn$

Applying the Master Theorem

- $T(n) = 2T(n/2) + cn$

- $a = ???, b = ???, f(n) = ???$

Applying the Master Theorem

- $T(n) = 2T(n/2) + cn$

- $a = 2$, $b = 2$, $f(n) = cn$

Applying the Master Theorem

- T(n) = 2T(n/2) + cn
- a = 2, b = 2, f(n) = cn
- Compare $n^{\log_b a}$ vs f(n)

Applying the Master Theorem

- T(n) = 2T(n/2) + cn
- a = 2, b = 2, f(n) = cn
- Compare $n^{\log_2 2}$ vs cn

Applying the Master Theorem

- T(n) = 2T(n/2) + cn
- a = 2, b = 2, f(n) = cn
- Compare n^1 vs cn

Applying the Master Theorem

- T(n) = 2T(n/2) + cn
- a = 2, b = 2, f(n) = cn
- Compare n^1 vs cn → f(n) = Θ($n^{\log_b a}$)
- Therefore T(n) = Θ(f(n) log n) = Θ(n log n)

Applying the Master Theorem

- $T(n) = T(2n/3) + c$

Applying the Master Theorem

- $T(n) = T(2n/3) + c$

- a = ???, b = ???, f(n) = ???

Applying the Master Theorem

- $T(n) = T(2n/3) + c$

- $a = 1$, $b = 3/2$, $f(n) = c$

Applying the Master Theorem

- T(n) = T(2n/3) + c
- a = 1, b = 3/2, f(n) = c
- Compare $n^{\log_b a}$ vs f(n)

Applying the Master Theorem

- T(n) = T(2n/3) + c
- a = 1, b = 3/2, f(n) = c
- Compare $n^{\log_{3/2} 1}$ vs cn^0

Applying the Master Theorem

- $T(n) = T(2n/3) + c$
- $a = 1$, $b = 3/2$, $f(n) = c$
- Compare n^0 vs cn^0 → $f(n) = \Theta(n^{\log_b a})$
- Therefore $T(n) = \Theta(f(n) \log n) = \Theta(\log n)$

Applying the Master Theorem

- $T(n) = 4T(n/2) + cn$

Applying the Master Theorem

- $T(n) = 4T(n/2) + cn$

- a = ???, b = ???, $f(n)$ = ???

Applying the Master Theorem

- $T(n) = 4T(n/2) + cn$

- $a = 4$, $b = 2$, $f(n) = cn$

Applying the Master Theorem

- T(n) = 4T(n/2) + cn
- a = 4, b = 2, f(n) = cn
- Compare $n^{\log_b a}$ vs f(n)

Applying the Master Theorem

- $T(n) = 4T(n/2) + cn$

- $a = 4$, $b = 2$, $f(n) = cn$

- Compare $n^{\log_2 4}$ vs cn

Applying the Master Theorem

- T(n) = 4T(n/2) + cn
- a = 4, b = 2, f(n) = cn
- Compare n^2 vs cn

Applying the Master Theorem

- T(n) = 4T(n/2) + cn
- a = 4, b = 2, f(n) = cn
- Compare n^2 vs cn → f(n) = O($n^{\log_b a - 1}$)
- Therefore T(n) = Θ($n^{\log_b a}$) = Θ(n^2)

Applying the Master Theorem

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

Applying the Master Theorem

- T(n) = 3T(n/4) + cn log n

- a = ???, b = ???, f(n) = ???

Applying the Master Theorem

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

- $a = 3$, $b = 4$, $f(n) = cn \log n$

Applying the Master Theorem

- T(n) = 3T(n/4) + cn log n
- a = 3, b = 4, f(n) = cn log n
- Compare $n^{\log_b a}$ vs f(n)

Applying the Master Theorem

- T(n) = 3T(n/4) + cn log n
- a = 3, b = 4, f(n) = cn log n
- Compare $n^{\log_4 3}$ vs cn log n

Applying the Master Theorem

- T(n) = 3T(n/4) + cn log n
- a = 3, b = 4, f(n) = cn log n
- Compare $n^{\log_4 3}$ vs cn log n → f(n) = $\Omega(n^{\log_b a+\varepsilon})$
- Therefore T(n) = Θ(f(n)) = Θ(n log n)

You’ll get more Master Method practice, *plus bonus experience with Binary Search*, in Studio 5.