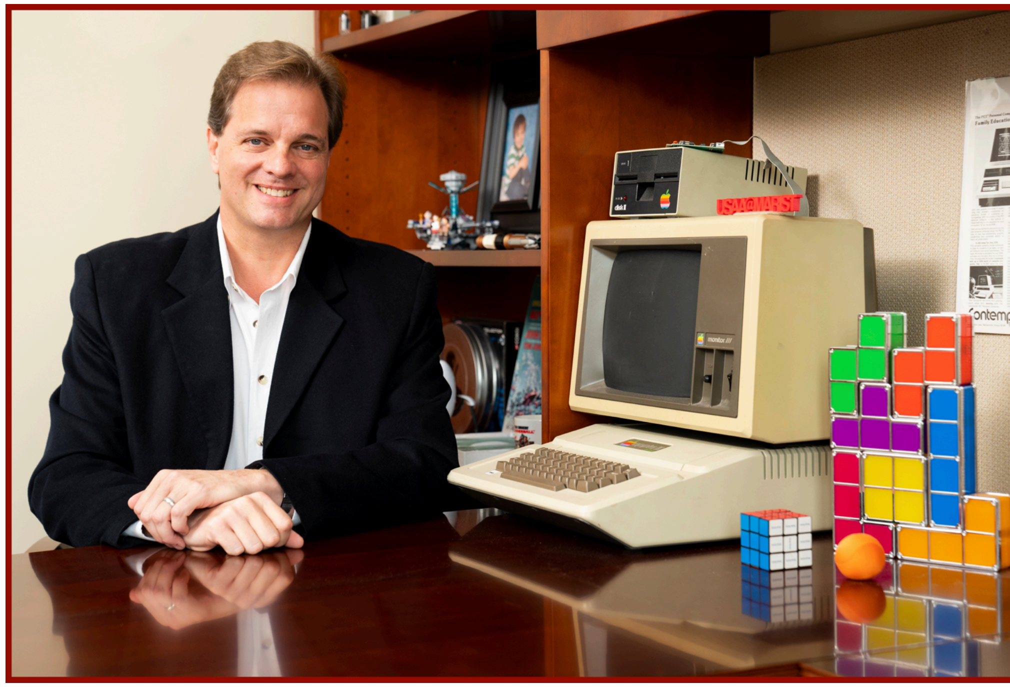
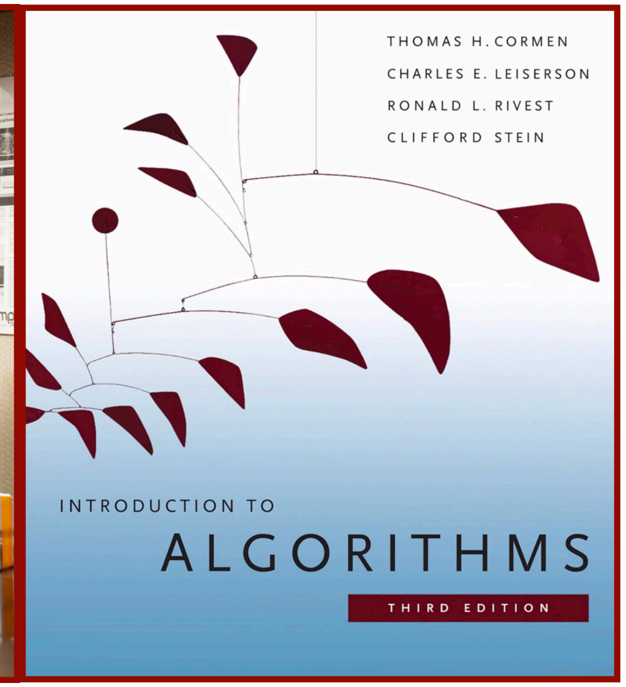
The Master Method





Alan G. Labouseur, Ph.D.
Alan.Labouseur@Marist.edu

Common Recurrences in Computer Science

Recurrences for algorithms we've implemented this semester.

$T(n) = T(n\text{-}1) + O(1)$	$O(n)$
$T(n) = T(n\text{-}1) + O(n)$	$O(n^2)$
$T(n) = T(\frac{n}{2}) + O(1)$	$O(\log_2 n)$
$T(n) = 2T(\frac{n}{2}) + O(n)$	$O(n \log_2 n)$

Common Recurrences in Computer Science

Recurrences for algorithms we've implemented this semester.

$T(n) = T(n\text{-}1) + O(1)$	$O(n)$	linear search, list traversal
$T(n) = T(n\text{-}1) + O(n)$	$O(n^2)$	
$T(n) = T(\frac{n}{2}) + O(1)$	$O(\log_2 n)$	
$T(n) = 2T(\frac{n}{2}) + O(n)$	$O(n \log_2 n)$	

Common Recurrences in Computer Science

Recurrences for algorithms we've implemented this semester.

$T(n) = T(n\text{-}1) + O(1)$	$O(n)$	
$T(n) = T(n\text{-}1) + O(n)$	$O(n^2)$	selection, insertion sort
$T(n) = T(\frac{n}{2}) + O(1)$	$O(\log_2 n)$	
$T(n) = 2T(\frac{n}{2}) + O(n)$	$O(n \log_2 n)$	

Common Recurrences in Computer Science

Recurrences for algorithms we've implemented this semester.

$T(n) = T(n\text{-}1) + O(1)$	$O(n)$	
$T(n) = T(n\text{-}1) + O(n)$	$O(n^2)$	
$T(n) = T(\frac{n}{2}) + O(1)$	$O(\log_2 n)$	binary search
$T(n) = 2T(\frac{n}{2}) + O(n)$	$O(n \log_2 n)$	

Common Recurrences in Computer Science

Recurrences for algorithms we've implemented this semester.

$T(n) = T(n-1) + O(1)$ O(n)

$T(n) = T(n-1) + O(n)$ $O(n^2)$

$T(n) = T(\frac{n}{2}) + O(1)$ $O(\log_2 n)$

$T(n) = 2T(\frac{n}{2}) + O(n)$ $O(n \log_2 n)$ quicksort, merge sort

Common Recurrences in Computer Science

Recurrences for algorithms we've implemented this semester.

$T(n) = T(n\text{-}1) + O(1)$	$O(n)$	linear search, list traversal
$T(n) = T(n\text{-}1) + O(n)$	$O(n^2)$	selection, insertion sort
$T(n) = T(\frac{n}{2}) + O(1)$	$O(\log_2 n)$	binary search
$T(n) = 2T(\frac{n}{2}) + O(n)$	$O(n \log_2 n)$	quicksort, merge sort

We can solve these with recursion trees or substitution. But is there are pattern here? Can this be generalized somehow?

Common Recurrences in Computer Science

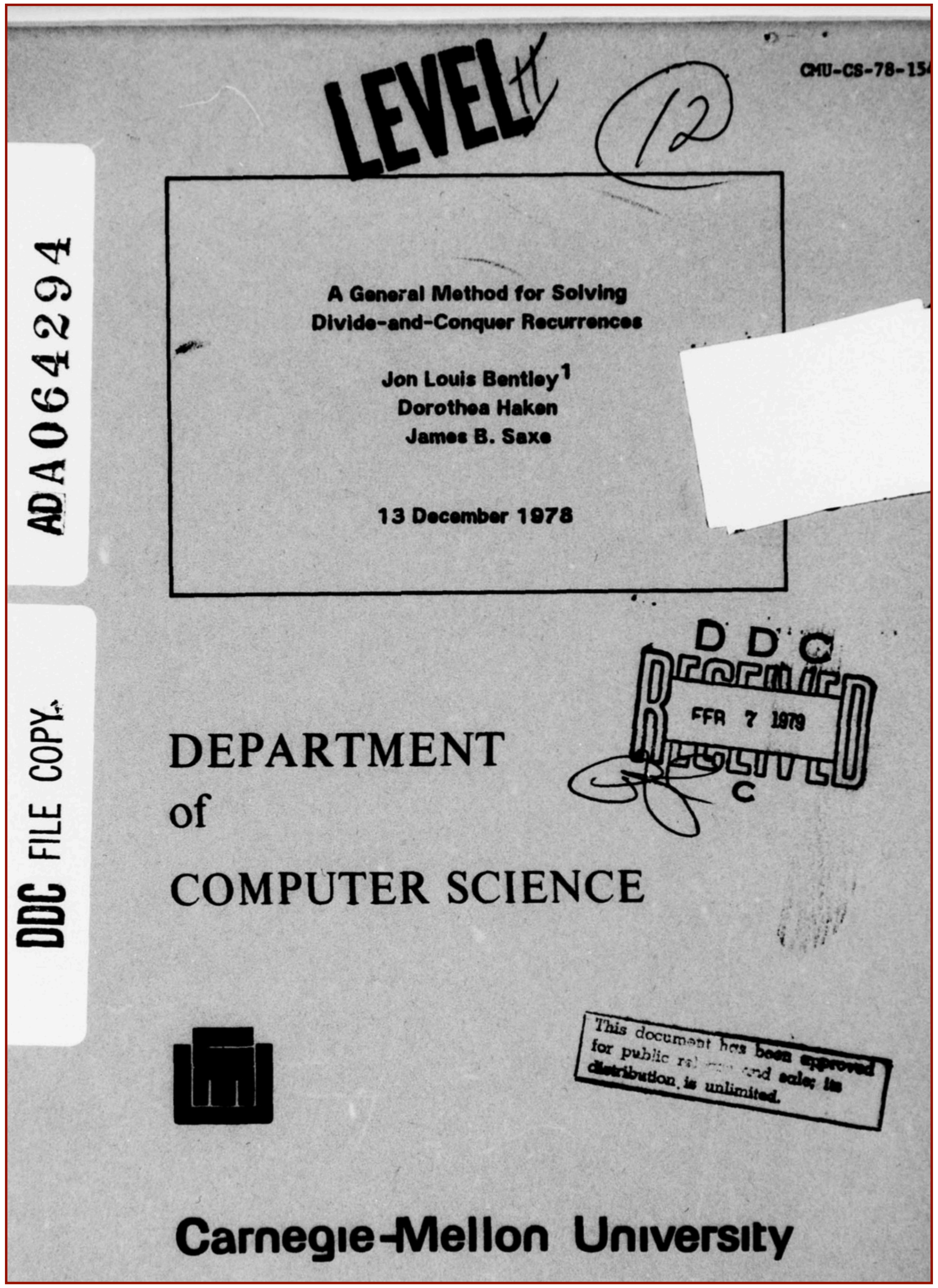
Recurrences for algorithms we've implemented this semester.

$T(n) = T(n\text{-}1) + O(1)$	$O(n)$	linear search, list traversal
$T(n) = T(n\text{-}1) + O(n)$	$O(n^2)$	selection, insertion sort
$T(n) = T(\frac{n}{2}) + O(1)$	$O(\log_2 n)$	binary search
$T(n) = 2T(\frac{n}{2}) + O(n)$	$O(n \log_2 n)$	quicksort, merge sort

Is there are pattern here? Can this be generalized somehow?

Jon Bentley saw the pattern for Divide and Conquer algorithms and generalized it.

The Master Theorem



CMU-CS-78-154

LEVEL II

12

ADA064294

DDC FILE COPY

A General Method for Solving
Divide-and-Conquer Recurrences

Jon Louis Bentley[1]
Dorothea Haken
James B. Saxe

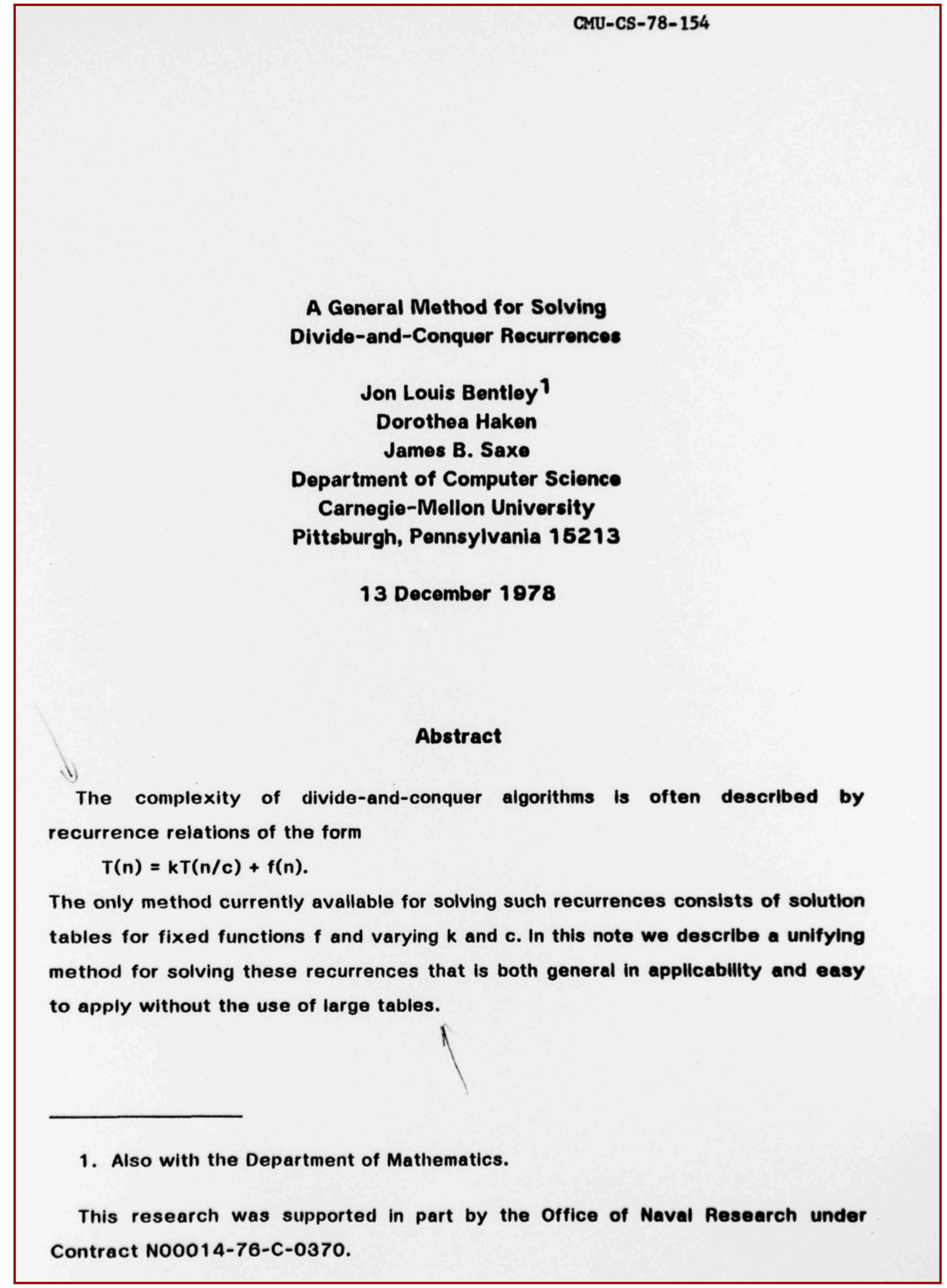
13 December 1978

DDC
RECEIVED
FEB 7 1979
C

DEPARTMENT
of
COMPUTER SCIENCE

This document has been approved for public release and sale; its distribution is unlimited.

Carnegie-Mellon University



CMU-CS-78-154

A General Method for Solving
Divide-and-Conquer Recurrences

Jon Louis Bentley[1]
Dorothea Haken
James B. Saxe
Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

13 December 1978

Abstract

The complexity of divide-and-conquer algorithms is often described by recurrence relations of the form

$T(n) = kT(n/c) + f(n).$

The only method currently available for solving such recurrences consists of solution tables for fixed functions f and varying k and c. In this note we describe a unifying method for solving these recurrences that is both general in applicability and easy to apply without the use of large tables.

1. Also with the Department of Mathematics.

This research was supported in part by the Office of Naval Research under Contract N00014-76-C-0370.

The Master Theorem

Given a recurrence in the form

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

where $a \geq 1$ and $b > 1$ and $f(n)$ is positive, there are three cases:

1. if $f(n) = O(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a})$
2. if $f(n) = \Theta(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a} \log_2 n)$
3. if $f(n) = \Omega(n^{\log_b a})$ then $T(n) = \Theta(f(n))$

The Master Theorem

Given a recurrence in the form

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

where $a \geq 1$ and $b > 1$ and $f(n)$ is positive, there are three cases:

1. if $f(n) = O(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a})$
2. if $f(n) = \Theta(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a} \log_2 n)$
3. if $f(n) = \Omega(n^{\log_b a})$ then $T(n) = \Theta(f(n))$

In each case, we compare $f(n)$ with $n^{\log_b a}$.

The Master Theorem

Given a recurrence in the form

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

where $a \geq 1$ and $b > 1$ and $f(n)$ is positive, there are three cases:

1. if $f(n) = O(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a})$
2. if $f(n) = \Theta(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a} \log_2 n)$
3. if $f(n) = \Omega(n^{\log_b a})$ then $T(n) = \Theta(f(n))$

In each case, we compare $f(n)$ with $n^{\log_b a}$.
Case 1 occurs when $f(n)$ is upper-bound by $n^{\log_b a}$.
We can think of this (roughly) as $f(n) < n^{\log_b a}$.
In this case the effort is dominated by $n^{\log_b a}$.

Specifically, $f(n)$ is **polynomially** smaller than $n^{\log_b a}$.

The Master Theorem

Given a recurrence in the form

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

where $a \geq 1$ and $b > 1$ and $f(n)$ is positive, there are three cases:

1. if $f(n) = O(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a})$
2. if $f(n) = \Theta(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a} \log_2 n)$
3. if $f(n) = \Omega(n^{\log_b a})$ then $T(n) = \Theta(f(n))$

In each case, we compare $f(n)$ with $n^{\log_b a}$.
Case 2 occurs when $f(n)$ is tight bound with $n^{\log_b a}$.
We can think of this (roughly) as $f(n) = n^{\log_b a}$.
In this case the effort is shared by $f(n)$ and $n^{\log_b a}$ so we multiply by a logarithmic factor (because of the height of the recursion tree).

The Master Theorem

Given a recurrence in the form

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

where $a \geq 1$ and $b > 1$ and $f(n)$ is positive, there are three cases:

1. if $f(n) = O(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a})$
2. if $f(n) = \Theta(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a} \log_2 n)$
3. if $f(n) = \Omega(n^{\log_b a})$ then $T(n) = \Theta(f(n))$

In each case, we compare $f(n)$ with $n^{\log_b a}$.
Case 3 occurs when $f(n)$ is lower bound by $n^{\log_b a}$.
We can think of this (roughly) as $f(n) > n^{\log_b a}$.
In this case the effort is dominated by $f(n)$.

Specifically, $f(n)$ is **polynomially** larger than $n^{\log_b a}$.

The Master Theorem

Given a recurrence in the form

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

where $a \geq 1$ and $b > 1$ and $f(n)$ is positive, there are three cases:

1. if $f(n) = O(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a})$
2. if $f(n) = \Theta(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a} \log_2 n)$
3. if $f(n) = \Omega(n^{\log_b a})$ then $T(n) = \Theta(f(n))$

One more requirement:
The sub-problems in these Divide and Conquer algorithms must be of equal size.

Even then, the Master Theorem does not always apply.

The Master Theorem

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

where $a \geq 1$ and $b > 1$ and $f(n)$ is positive, there are three cases:

1. if $f(n) = O(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a})$
2. if $f(n) = \Theta(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a} \log_2 n)$
3. if $f(n) = \Omega(n^{\log_b a})$ then $T(n) = \Theta(f(n))$

Example: $T(n) = T\left(\frac{n}{2}\right) + O(1)$

The Master Theorem

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

where $a \geq 1$ and $b > 1$ and $f(n)$ is positive, there are three cases:

1. if $f(n) = O(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a})$
2. if $f(n) = \Theta(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a} \log_2 n)$
3. if $f(n) = \Omega(n^{\log_b a})$ then $T(n) = \Theta(f(n))$

Example: $T(n) = T\left(\frac{n}{2}\right) + O(1)$

$$= 1T\left(\frac{n}{2}\right) + O(1)$$

$a = 1$

$b = 2$

$f(n) = 1$

The Master Theorem

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

where $a \geq 1$ and $b > 1$ and $f(n)$ is positive, there are three cases:

1. if $f(n) = O(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a})$
2. if $f(n) = \Theta(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a} \log_2 n)$
3. if $f(n) = \Omega(n^{\log_b a})$ then $T(n) = \Theta(f(n))$

Example: $T(n) = T\left(\frac{n}{2}\right) + O(1)$

$$= 1T\left(\frac{n}{2}\right) + O(1)$$

$a = 1$

$b = 2$

$f(n) = 1$

The Master Theorem

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

where $a \geq 1$ and $b > 1$ and $f(n)$ is positive, there are three cases:

1. if $f(n) = O(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a})$
2. if $f(n) = \Theta(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a} \log_2 n)$
3. if $f(n) = \Omega(n^{\log_b a})$ then $T(n) = \Theta(f(n))$

Example: $T(n) = T\left(\frac{n}{2}\right) + O(1)$

$$= 1T\left(\frac{n}{2}\right) + O(1)$$

$a = 1$

$b = 2$

$f(n) = 1$

compute $n^{\log_b a} = n^{\log_2 1}$

$= n^0$ (because $2^0=1$)

$= 1$

The Master Theorem

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

where $a \geq 1$ and $b > 1$ and $f(n)$ is positive, there are three cases:

1. if $f(n) = O(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a})$
2. if $f(n) = \Theta(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a} \log_2 n)$
3. if $f(n) = \Omega(n^{\log_b a})$ then $T(n) = \Theta(f(n))$

Example: $T(n) = T\left(\frac{n}{2}\right) + O(1)$

$$= 1T\left(\frac{n}{2}\right) + O(1)$$

$a = 1$ compute $n^{\log_b a} = n^{\log_2 1}$

$b = 2$ $= n^0$ (because $2^0=1$)

$f(n) = 1$ ⟵ compare ⟶ $= 1$

The Master Theorem

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

where $a \geq 1$ and $b > 1$ and $f(n)$ is positive, there are three cases:

1. if $f(n) = O(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a})$
2. if $f(n) = \Theta(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a} \log_2 n)$
3. if $f(n) = \Omega(n^{\log_b a})$ then $T(n) = \Theta(f(n))$

Example: $T(n) = T\left(\frac{n}{2}\right) + O(1)$

$$= 1T\left(\frac{n}{2}\right) + O(1)$$

$a = 1$

$b = 2$

$f(n) = 1$ ⟵ Equal. Case 2 ⟶ $= 1$

compute $n^{\log_b a} = n^{\log_2 1}$

$= n^0$ (because $2^0=1$)

$= 1$

The Master Theorem

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

where $a \geq 1$ and $b > 1$ and $f(n)$ is positive, there are three cases:

1. if $f(n) = O(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a})$
2. if $f(n) = \Theta(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a} \log_2 n)$
3. if $f(n) = \Omega(n^{\log_b a})$ then $T(n) = \Theta(f(n))$

Example: $T(n) = T\left(\frac{n}{2}\right) + O(1)$

$$= 1T\left(\frac{n}{2}\right) + O(1)$$

$a = 1$

$b = 2$

$f(n) = 1$ ⟵ Equal. Case 2 ⟶

compute $n^{\log_b a} = n^{\log_2 1}$

$= n^0$ (because $2^0=1$)

$= 1$

$$T(n) = \Theta(n^{\log_b a} \log_2 n)$$

$$= \Theta(1 \log_2 n)$$

$$= \Theta(\log_2 n)$$

The Master Theorem

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

where $a \geq 1$ and $b > 1$ and $f(n)$ is positive, there are three cases:

1. if $f(n) = O(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a})$
2. if $f(n) = \Theta(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a} \log_2 n)$
3. if $f(n) = \Omega(n^{\log_b a})$ then $T(n) = \Theta(f(n))$

Example: $T(n) = T\left(\frac{n}{2}\right) + O(1)$

$$= 1T\left(\frac{n}{2}\right) + O(1)$$

$a = 1$

$b = 2$

$f(n) = 1$ ⟵ Equal. Case 2 ⟶ $= 1$

compute $n^{\log_b a} = n^{\log_2 1}$

$= n^0$ (because $2^0=1$)

$= 1$

$$T(n) = \Theta(n^{\log_b a} \log_2 n)$$

$$= \Theta(1 \log_2 n)$$

Binary Search is $\Theta(\log_2 n)$

The Master Theorem

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

where $a \geq 1$ and $b > 1$ and $f(n)$ is positive, there are three cases:

1. if $f(n) = O(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a})$
2. if $f(n) = \Theta(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a} \log_2 n)$
3. if $f(n) = \Omega(n^{\log_b a})$ then $T(n) = \Theta(f(n))$

Example: $T(n) = 2T\left(\frac{n}{2}\right) + O(n)$

$a = 2$

$b = 2$

$f(n) = n$

The Master Theorem

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

where $a \geq 1$ and $b > 1$ and $f(n)$ is positive, there are three cases:

1. if $f(n) = O(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a})$
2. if $f(n) = \Theta(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a} \log_2 n)$
3. if $f(n) = \Omega(n^{\log_b a})$ then $T(n) = \Theta(f(n))$

Example: $T(n) = 2T\left(\frac{n}{2}\right) + O(n)$

$a = 2$

$b = 2$

$f(n) = n$

compute $n^{\log_b a} = n^{\log_2 2}$

$= n^1$

$= n$

The Master Theorem

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

where $a \geq 1$ and $b > 1$ and $f(n)$ is positive, there are three cases:

1. if $f(n) = O(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a})$
2. if $f(n) = \Theta(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a} \log_2 n)$
3. if $f(n) = \Omega(n^{\log_b a})$ then $T(n) = \Theta(f(n))$

Example: $T(n) = 2T\left(\frac{n}{2}\right) + O(n)$

$a = 2$

$b = 2$

$f(n) = n$ ⟵ Equal. Case 2 ⟶ $= n$

compute $n^{\log_b a} = n^{\log_2 2} = n^1 = n$

$$T(n) = \Theta(n^{\log_b a} \log_2 n) = \Theta(n \log_2 n)$$

Merge sort.

The Master Theorem

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

where $a \geq 1$ and $b > 1$ and $f(n)$ is positive, there are three cases:

1. if $f(n) = O(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a})$
2. if $f(n) = \Theta(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a} \log_2 n)$
3. if $f(n) = \Omega(n^{\log_b a})$ then $T(n) = \Theta(f(n))$

Example: $T(n) = 2T\left(\frac{n}{4}\right) + O(1)$

$a = 2$

$b = 4$

$f(n) = 1$

compute $n^{\log_b a} = n^{\log_4 2}$

$= n^{1/2}$ (because $4^{1/2}=2$)

$= \sqrt{n}$

The Master Theorem

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

where $a \geq 1$ and $b > 1$ and $f(n)$ is positive, there are three cases:

1. if $f(n) = O(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a})$
2. if $f(n) = \Theta(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a} \log_2 n)$
3. if $f(n) = \Omega(n^{\log_b a})$ then $T(n) = \Theta(f(n))$

Example: $T(n) = 2T\left(\frac{n}{4}\right) + O(1)$

$a = 2$

$b = 4$

$f(n) = 1 \longleftarrow$ compare $\longrightarrow = \sqrt{n}$

compute $n^{\log_b a} = n^{\log_4 2}$

$= n^{1/2}$ (because $4^{1/2}=2$)

$= \sqrt{n}$

The Master Theorem

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

where $a \geq 1$ and $b > 1$ and $f(n)$ is positive, there are three cases:

1. if $f(n) = O(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a})$
2. if $f(n) = \Theta(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a} \log_2 n)$
3. if $f(n) = \Omega(n^{\log_b a})$ then $T(n) = \Theta(f(n))$

Example: $T(n) = 2T\left(\frac{n}{4}\right) + O(1)$

$a = 2$

$b = 4$

$f(n) = 1$

compute $n^{\log_b a} = n^{\log_4 2}$

$= n^{1/2}$ (because $4^{1/2}=2$)

$= \sqrt{n}$

$f(n) = 1 \longleftarrow 1 < \sqrt{n}$ for $n > 1 \longrightarrow = \sqrt{n}$

Case 1

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

The Master Theorem

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

where $a \geq 1$ and $b > 1$ and $f(n)$ is positive, there are three cases:

1. if $f(n) = O(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a})$
2. if $f(n) = \Theta(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a} \log_2 n)$
3. if $f(n) = \Omega(n^{\log_b a})$ then $T(n) = \Theta(f(n))$

Example: $T(n) = 2T\left(\frac{n}{4}\right) + O(n)$

$a = 2$

$b = 4$

$f(n) = n$

compute $n^{\log_b a} = n^{\log_4 2}$

$= n^{1/2}$ (because $4^{1/2}=2$)

$= \sqrt{n}$

The Master Theorem

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

where $a \geq 1$ and $b > 1$ and $f(n)$ is positive, there are three cases:

1. if $f(n) = O(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a})$
2. if $f(n) = \Theta(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a} \log_2 n)$
3. if $f(n) = \Omega(n^{\log_b a})$ then $T(n) = \Theta(f(n))$

Example: $T(n) = 2T\left(\frac{n}{4}\right) + O(n)$

$a = 2$

$b = 4$

$f(n) = n$

compute $n^{\log_b a} = n^{\log_4 2}$

$= n^{1/2}$ (because $4^{1/2}=2$)

$= \sqrt{n}$

$n > \sqrt{n}$

Case 3

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

$$= \Theta(n)$$

The Master Theorem

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

where $a \geq 1$ and $b > 1$ and $f(n)$ is positive, there are three cases:

1. if $f(n) = O(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a})$
2. if $f(n) = \Theta(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a} \log_2 n)$
3. if $f(n) = \Omega(n^{\log_b a})$ then $T(n) = \Theta(f(n))$

Example: $T(n) = T(n\text{-}1) + O(n)$ — selection sort, so we expect $O(n^2)$

The Master Theorem

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

where $a \geq 1$ and $b > 1$ and $f(n)$ is positive, there are three cases:

1. if $f(n) = O(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a})$
2. if $f(n) = \Theta(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a} \log_2 n)$
3. if $f(n) = \Omega(n^{\log_b a})$ then $T(n) = \Theta(f(n))$

Example: $T(n) = T(n\text{-}1) + O(n)$ — selection sort, so we expect $O(n^2)$

$$= 1T\left(\frac{n-1}{1}\right) + O(n)$$

$a = 1$

$b = 1$

$f(n) = n$

compute $n^{\log_b a} = n^{\log_1 1}$

$= ?$

The Master Theorem

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

where $a \geq 1$ and $b > 1$ and $f(n)$ is positive, there are three cases:

1. if $f(n) = O(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a})$
2. if $f(n) = \Theta(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a} \log_2 n)$
3. if $f(n) = \Omega(n^{\log_b a})$ then $T(n) = \Theta(f(n))$

compute $n^{\log_b a} = n^{\log_1 1}$

$\log_1 1 = X$

means $1^X = 1$

so . . .

The Master Theorem

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

where $a \geq 1$ and $b > 1$ and $f(n)$ is positive, there are three cases:

1. if $f(n) = O(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a})$
2. if $f(n) = \Theta(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a} \log_2 n)$
3. if $f(n) = \Omega(n^{\log_b a})$ then $T(n) = \Theta(f(n))$

compute $n^{\log_b a} = n^{\log_1 1}$

$\log_1 1 = X$

means $1^X = 1$
so X = anything

The Master Theorem

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

where $a \geq 1$ and $b > 1$ and $f(n)$ is positive, there are three cases:

1. if $f(n) = O(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a})$
2. if $f(n) = \Theta(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a} \log_2 n)$
3. if $f(n) = \Omega(n^{\log_b a})$ then $T(n) = \Theta(f(n))$

Example: $T(n) = T(n\text{-}1) + O(n)$ — selection sort, so we expect $O(n^2)$

$$= 1T\left(\frac{n-1}{1}\right) + O(n)$$

$a = 1$

$b = 1$

$f(n) = n$

compute $n^{\log_b a} = n^{\log_1 1}$

$= n^{\text{anything?}}$

$= n^{\text{nothing!}}$

= undefined at best

= division by 0 at worst

The Master Theorem

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

where $a \geq 1$ and $b > 1$ and $f(n)$ is positive, there are three cases:

1. if $f(n) = O(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a})$
2. if $f(n) = \Theta(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a} \log_2 n)$
3. if $f(n) = \Omega(n^{\log_b a})$ then $T(n) = \Theta(f(n))$

Example: $T(n) = T(n-1) + O(n)$ — selection sort, so we expect $O(n^2)$

$$= 1T\left(\frac{n-1}{1}\right) + O(n)$$

$a = 1$

$b = 1$

$f(n) = n$

compute $n^{\log_b a} = n^{\log_1 1}$

$= n^{\text{anything?}}$

$= n^{\text{nothing!}}$

= undefined at best

= division by 0 at worst

The Master Method does not apply to this recurrence.
Why not?

The Master Theorem

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

where $a \geq 1$ and $b > 1$ and $f(n)$ is positive, there are three cases:

1. if $f(n) = O(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a})$
2. if $f(n) = \Theta(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a} \log_2 n)$
3. if $f(n) = \Omega(n^{\log_b a})$ then $T(n) = \Theta(f(n))$

Example: $T(n) = T(n\text{-}1) + O(n)$ — selection sort, so we expect $O(n^2)$

$$= 1T\left(\frac{n\text{-}1}{1}\right) + O(n)$$

$a = 1$

$b = 1$

$f(n) = n$

compute $n^{\log_b a} = n^{\log_1 1}$

$= n^{\text{anything?}}$

$= n^{\text{nothing!}}$

= undefined at best

= division by 0 at worst

The Master Method does not apply to this recurrence.
Why not?

The Master Theorem

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

where $a \geq 1$ and $b > 1$ and $f(n)$ is positive, there are three cases:

1. if $f(n) = O(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a})$
2. if $f(n) = \Theta(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a} \log_2 n)$
3. if $f(n) = \Omega(n^{\log_b a})$ then $T(n) = \Theta(f(n))$

Example: $T(n) = T(n-1) + O(1)$ — linear search, so we expect $O(n)$

The Master Theorem

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

where $a \geq 1$ and $b > 1$ and $f(n)$ is positive, there are three cases:

1. if $f(n) = O(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a})$
2. if $f(n) = \Theta(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a} \log_2 n)$
3. if $f(n) = \Omega(n^{\log_b a})$ then $T(n) = \Theta(f(n))$

Example: $T(n) = T(n\text{-}1) + O(1)$ — linear search, so we expect O(n)

$$= 1T\left(\frac{n\text{-}1}{1}\right) + O(1)$$

$a = 1$

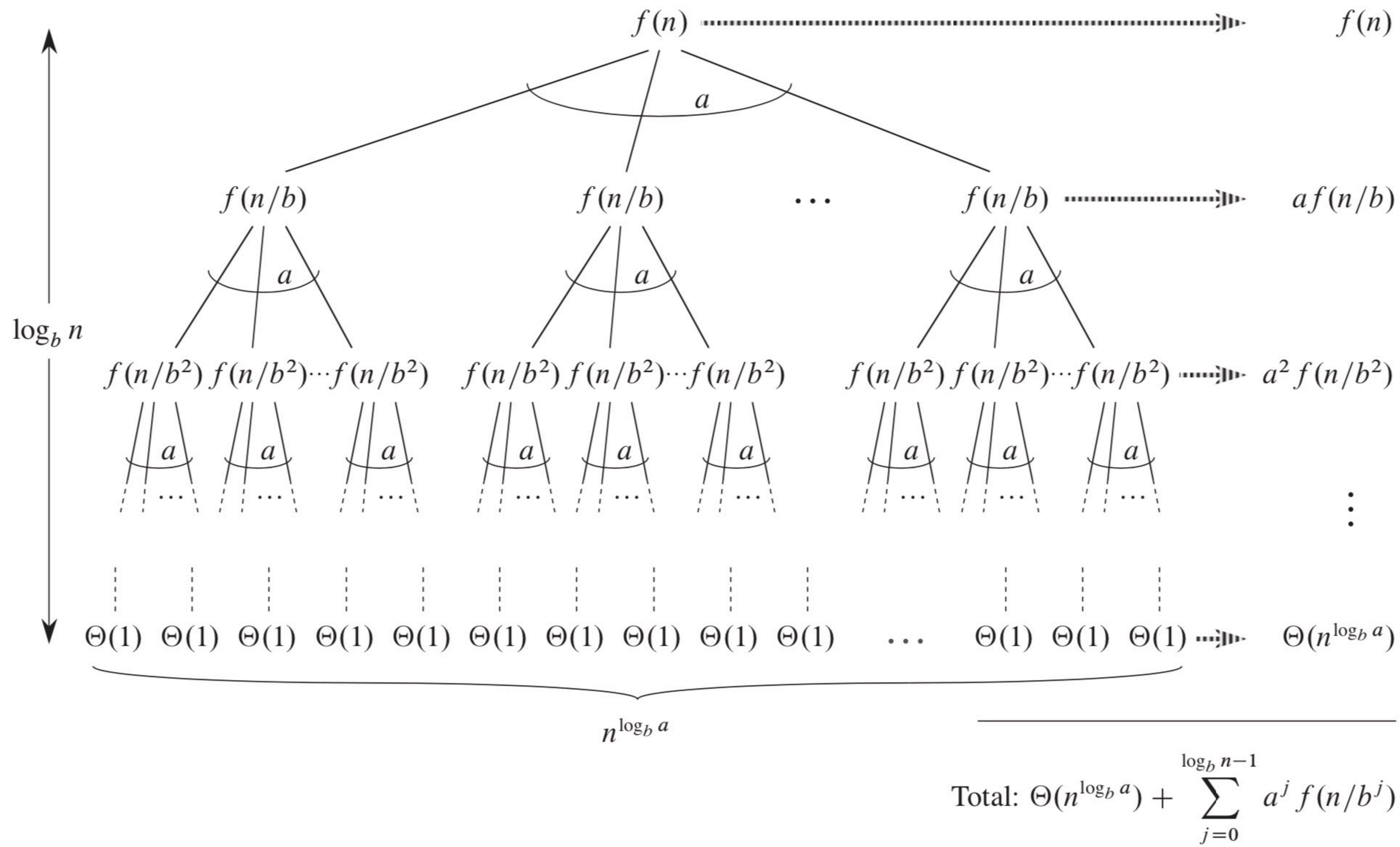
$b = 1$

$f(n) = 1$

The Master Method does not apply to this recurrence either.

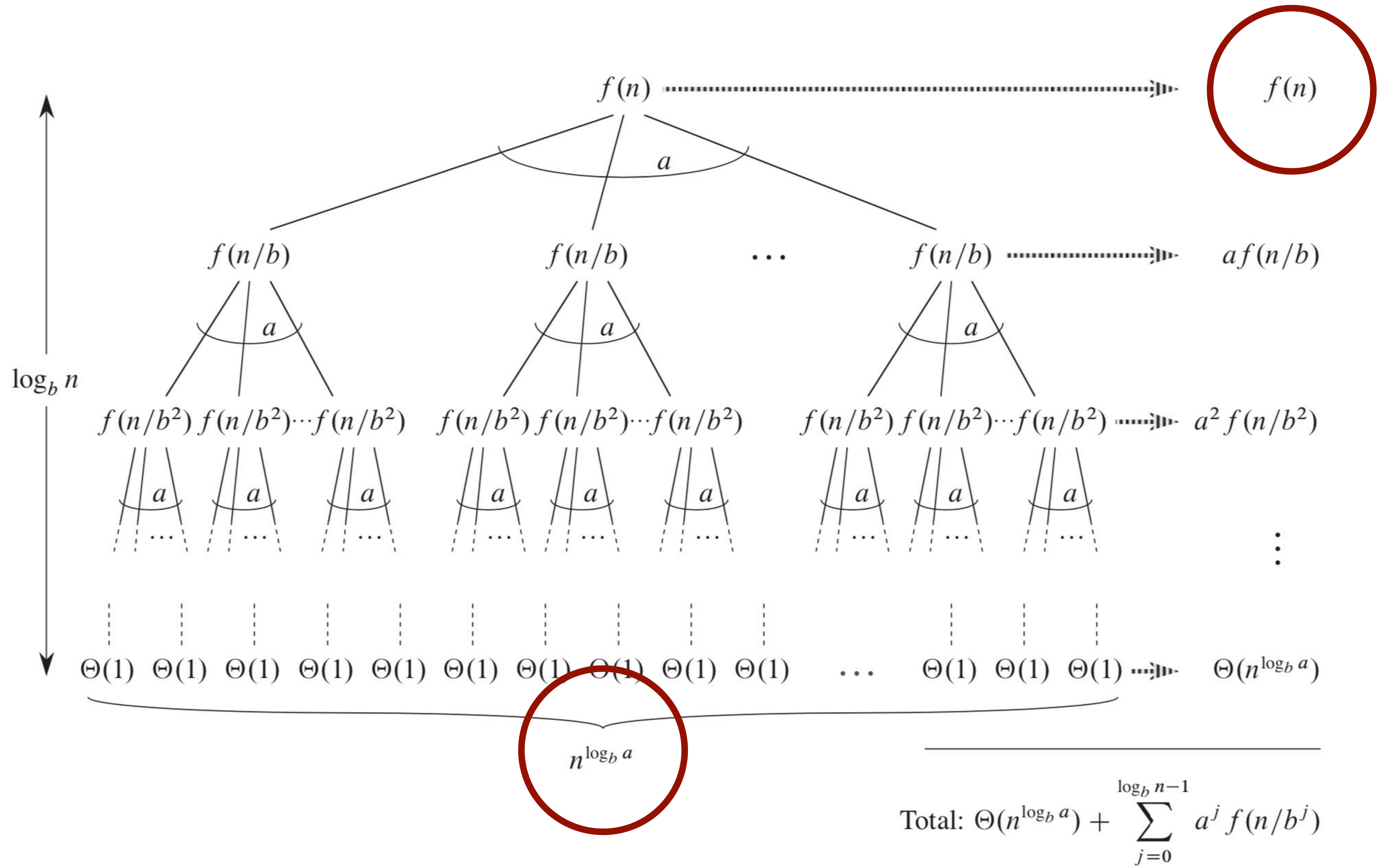
The Master Theorem

Why does this work? Remember recursion trees?



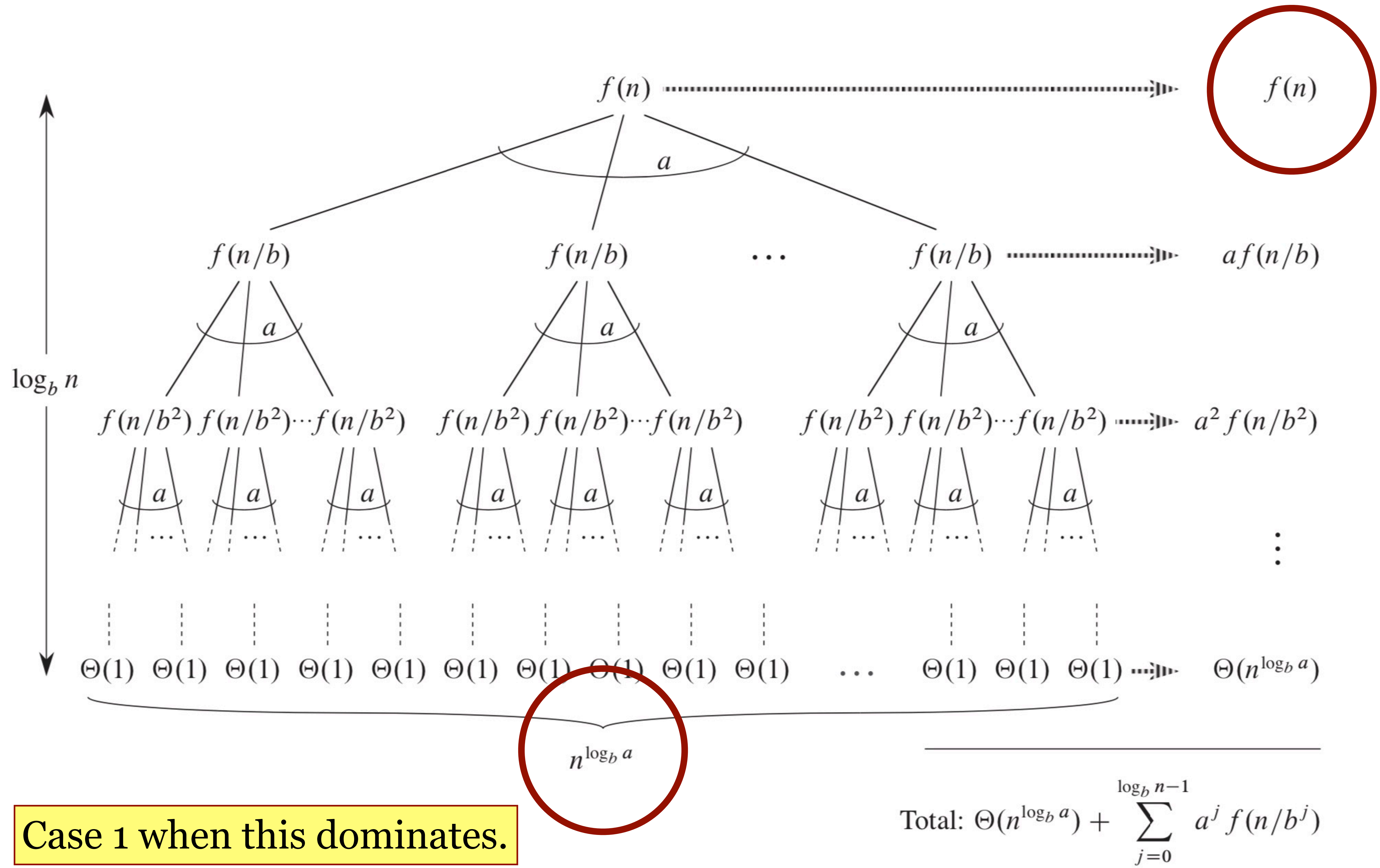
The Master Theorem

Why does this work? Remember recursion trees?



The Master Theorem

Why does this work? Remember recursion trees?

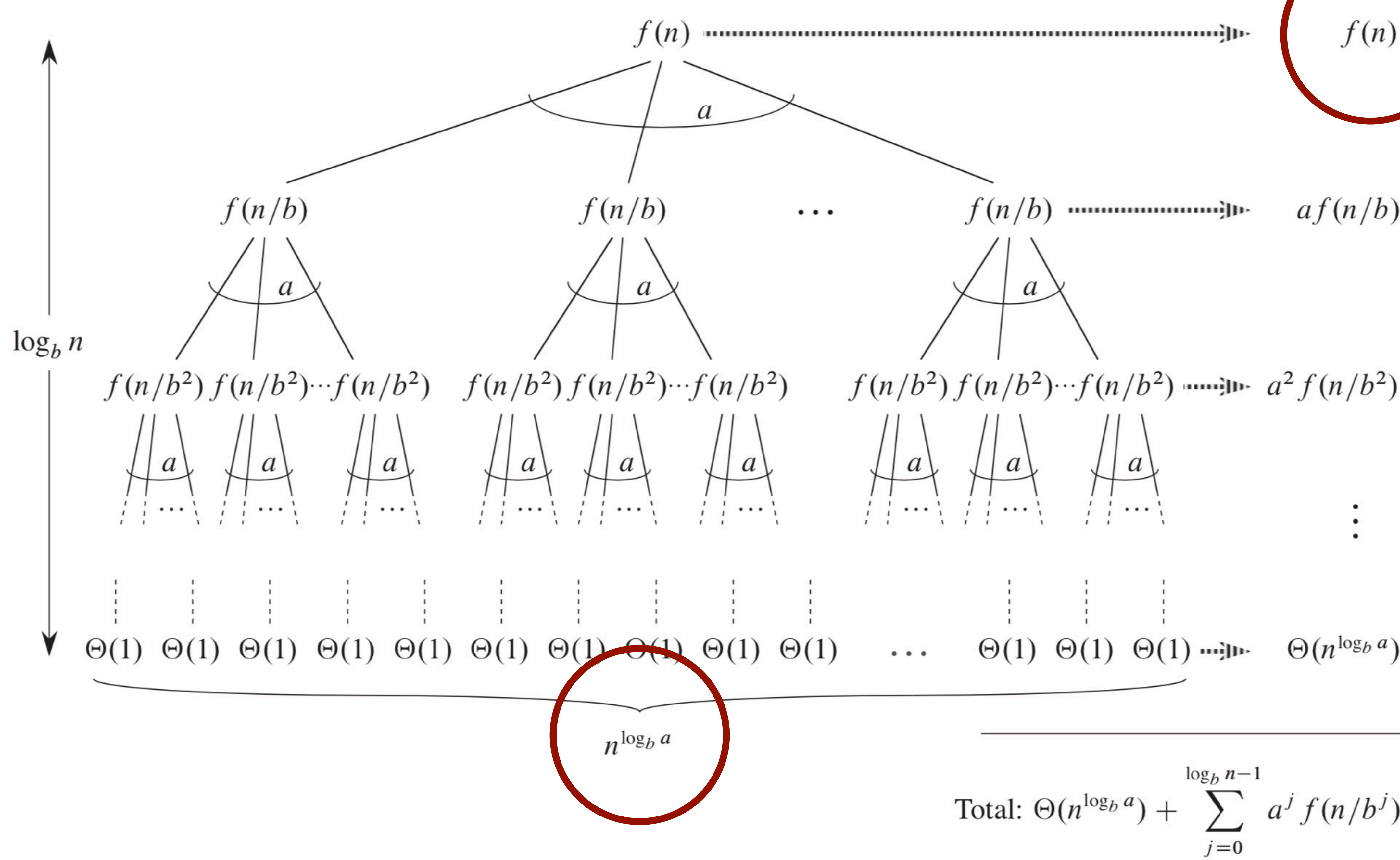


Case 1 when this dominates.

The Master Theorem

Why does this work? Remember recursion trees?

Case 3 when this dominates.



The Master Theorem

Why does this work? Remember recursion trees?