CSE 431/531: Algorithm Analysis and Design (Fall 2024)

Divide-and-Conquer

Lecturer: Kelin Luo

Department of Computer Science and Engineering
University at Buffalo

Outline

Outline

1. Divide-and-Conquer
2. Counting Inversions
3. Quicksort and Selection
   - Lower Bound for Comparison-Based Sorting Algorithms
   - Selection Problem
4. Polynomial Multiplication
5. Solving Recurrences
6. Computing n-th Fibonacci Number

Outline

Outline

Outline

Outline

Outline

Methods for Solving Recurrences

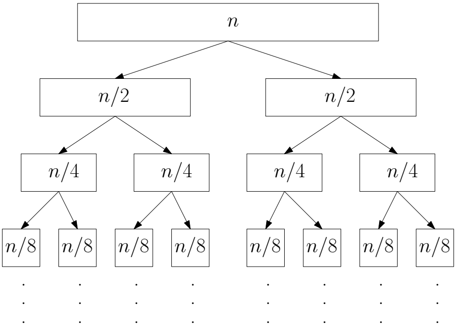
- The recursion-tree method
- The master theorem

Recursion-Tree Method

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

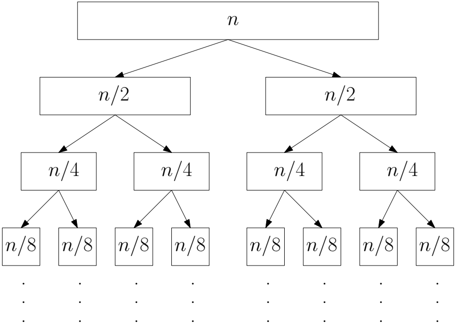
Recursion-Tree Method

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



Recursion-Tree Method

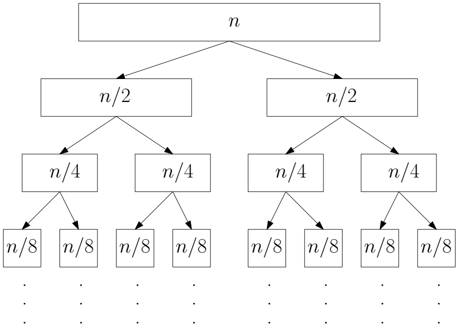
- $T(n) = 2T(n/2) + O(n)$



- Each level takes running time $O(n)$

Recursion-Tree Method

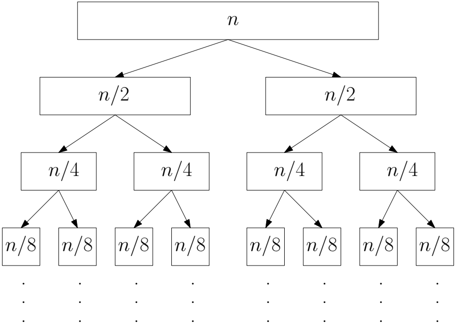
- $T(n) = 2T(n/2) + O(n)$



- Each level takes running time $O(n)$
- There are $O(\lg n)$ levels

Recursion-Tree Method

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



- Each level takes running time $O(n)$
- There are $O(\lg n)$ levels
- Running time $= O(n \lg n)$

Recursion-Tree Method

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

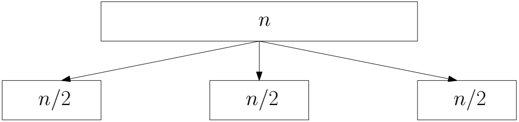
Recursion-Tree Method

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

$$\boxed{\quad n \quad}$$

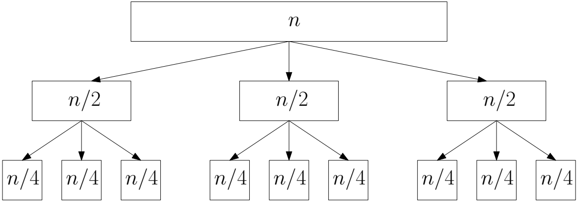
Recursion-Tree Method

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



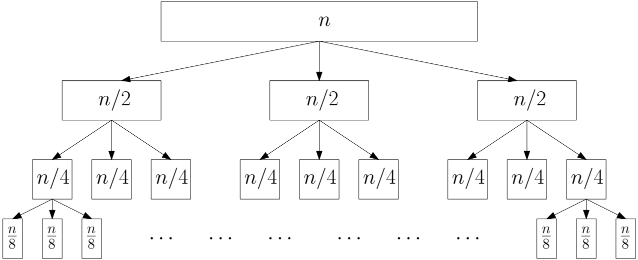
Recursion-Tree Method

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



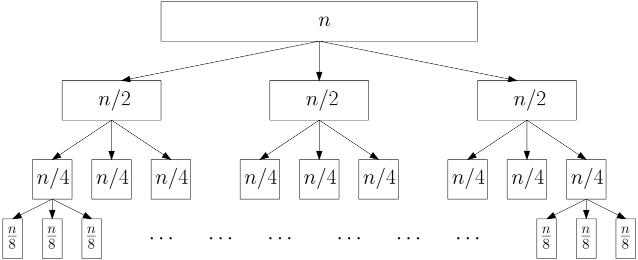
Recursion-Tree Method

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



Recursion-Tree Method

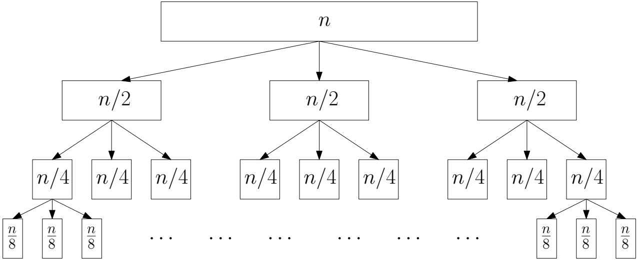
- $T(n) = 3T(n/2) + O(n)$



- Total running time at level i?

Recursion-Tree Method

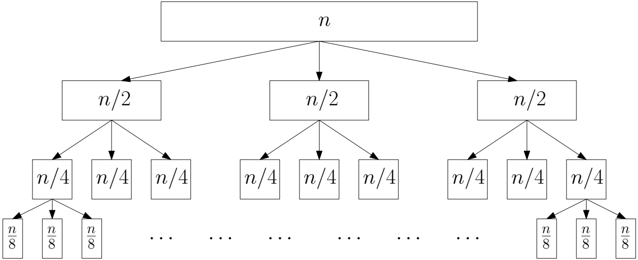
- $T(n) = 3T(n/2) + O(n)$



- Total running time at level i? $\frac{n}{2^i} \times 3^i = \left(\frac{3}{2}\right)^i n$

Recursion-Tree Method

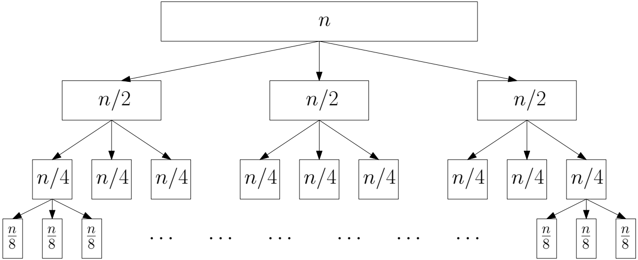
- $T(n) = 3T(n/2) + O(n)$



- Total running time at level i? $\frac{n}{2^i} \times 3^i = \left(\frac{3}{2}\right)^i n$
- Index of last level?

Recursion-Tree Method

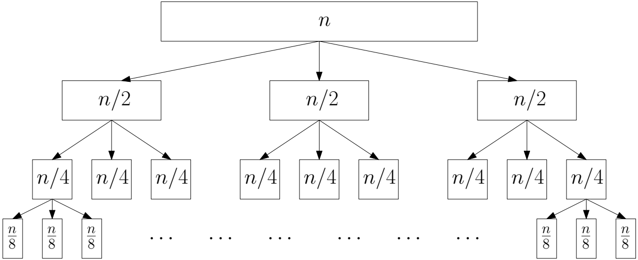
- $T(n) = 3T(n/2) + O(n)$



- Total running time at level i? $\frac{n}{2^i} \times 3^i = \left(\frac{3}{2}\right)^i n$
- Index of last level? $\lg_2 n$

Recursion-Tree Method

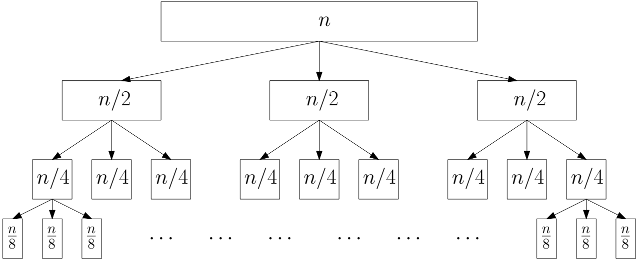
- $T(n) = 3T(n/2) + O(n)$



- Total running time at level i? $\frac{n}{2^i} \times 3^i = \left(\frac{3}{2}\right)^i n$
- Index of last level? $\lg_2 n$
- Total running time?

Recursion-Tree Method

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



- Total running time at level i? $\frac{n}{2^i} \times 3^i = \left(\frac{3}{2}\right)^i n$
- Index of last level? $\lg_2 n$
- Total running time?

$$\sum_{i=0}^{\lg_2 n} \left(\frac{3}{2}\right)^i n = O\left(n \left(\frac{3}{2}\right)^{\lg_2 n}\right) = O(3^{\lg_2 n}) = O(n^{\lg_2 3}).$$

Recursion-Tree Method

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

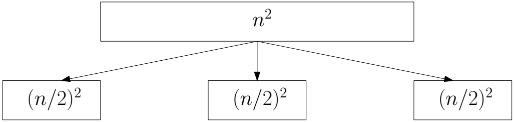
Recursion-Tree Method

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

$$\boxed{n^2}$$

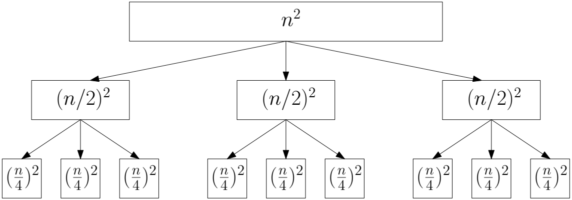
Recursion-Tree Method

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



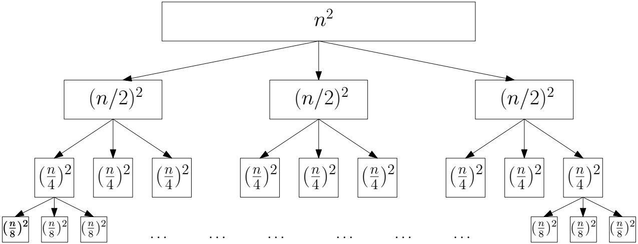
Recursion-Tree Method

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



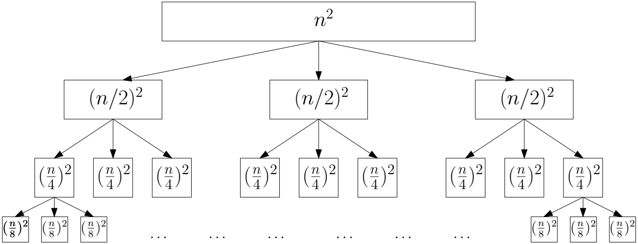
Recursion-Tree Method

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



Recursion-Tree Method

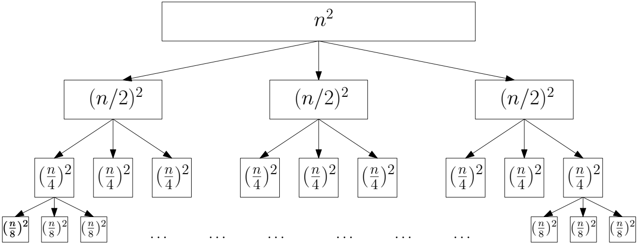
- $T(n) = 3T(n/2) + O(n^2)$



- Total running time at level i?

Recursion-Tree Method

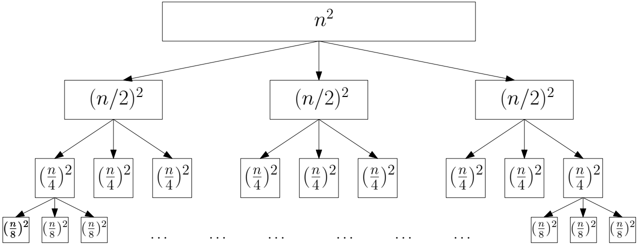
- $T(n) = 3T(n/2) + O(n^2)$



- Total running time at level i? $\left(\frac{n}{2^i}\right)^2 \times 3^i = \left(\frac{3}{4}\right)^i n^2$

Recursion-Tree Method

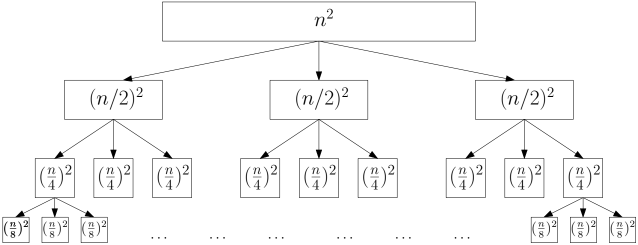
- $T(n) = 3T(n/2) + O(n^2)$



- Total running time at level i? $\left(\frac{n}{2^i}\right)^2 \times 3^i = \left(\frac{3}{4}\right)^i n^2$
- Index of last level?

Recursion-Tree Method

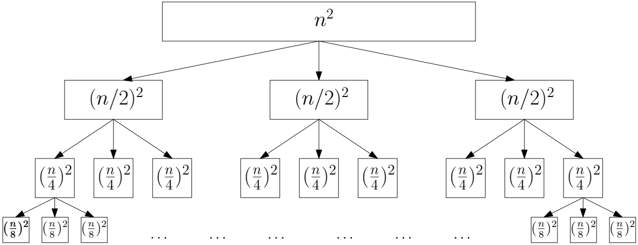
- $T(n) = 3T(n/2) + O(n^2)$



- Total running time at level i? $\left(\frac{n}{2^i}\right)^2 \times 3^i = \left(\frac{3}{4}\right)^i n^2$
- Index of last level? $\lg_2 n$

Recursion-Tree Method

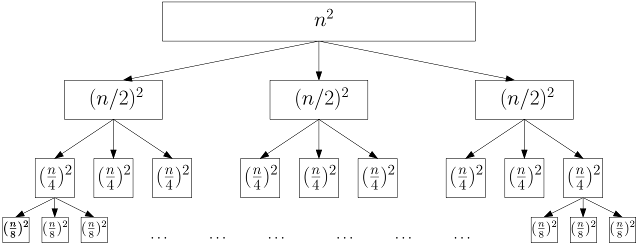
- $T(n) = 3T(n/2) + O(n^2)$



- Total running time at level i? $\left(\frac{n}{2^i}\right)^2 \times 3^i = \left(\frac{3}{4}\right)^i n^2$
- Index of last level? $\lg_2 n$
- Total running time?

Recursion-Tree Method

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

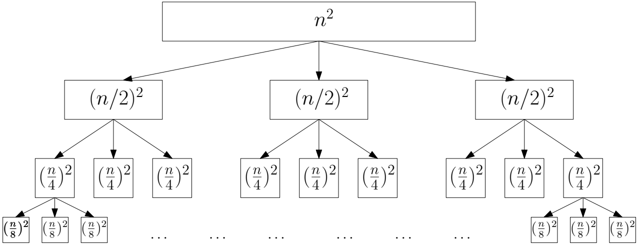


- Total running time at level i? $\left(\frac{n}{2^i}\right)^2 \times 3^i = \left(\frac{3}{4}\right)^i n^2$
- Index of last level? $\lg_2 n$
- Total running time?

$$\sum_{i=0}^{\lg_2 n} \left(\frac{3}{4}\right)^i n^2 =$$

Recursion-Tree Method

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



- Total running time at level i? $\left(\frac{n}{2^i}\right)^2 \times 3^i = \left(\frac{3}{4}\right)^i n^2$
- Index of last level? $\lg_2 n$
- Total running time?

$$\sum_{i=0}^{\lg_2 n} \left(\frac{3}{4}\right)^i n^2 = O(n^2).$$

Master Theorem

Recurrences	a	b	c	time
$T(n) = 2T(n/2) + O(n)$				$O(n \lg n)$
$T(n) = 3T(n/2) + O(n)$				$O(n^{\lg_2 3})$
$T(n) = 3T(n/2) + O(n^2)$				$O(n^2)$

Theorem $T(n) = aT(n/b) + O(n^c)$, where $a \geq 1, b > 1, c \geq 0$ are constants. Then,

Master Theorem

Recurrences	a	b	c	time
$T(n) = 2T(n/2) + O(n)$	2	2	1	$O(n \lg n)$
$T(n) = 3T(n/2) + O(n)$				$O(n^{\lg_2 3})$
$T(n) = 3T(n/2) + O(n^2)$				$O(n^2)$

Theorem $T(n) = aT(n/b) + O(n^c)$, where $a \geq 1, b > 1, c \geq 0$ are constants. Then,

Master Theorem

Recurrences	a	b	c	time
$T(n) = 2T(n/2) + O(n)$	2	2	1	$O(n \lg n)$
$T(n) = 3T(n/2) + O(n)$	3	2	1	$O(n^{\lg_2 3})$
$T(n) = 3T(n/2) + O(n^2)$				$O(n^2)$

Theorem $T(n) = aT(n/b) + O(n^c)$, where $a \geq 1, b > 1, c \geq 0$ are constants. Then,

Master Theorem

Recurrences	a	b	c	time
$T(n) = 2T(n/2) + O(n)$	2	2	1	$O(n \lg n)$
$T(n) = 3T(n/2) + O(n)$	3	2	1	$O(n^{\lg_2 3})$
$T(n) = 3T(n/2) + O(n^2)$	3	2	2	$O(n^2)$

Theorem $T(n) = aT(n/b) + O(n^c)$, where $a \geq 1, b > 1, c \geq 0$ are constants. Then,

Master Theorem

Recurrences	a	b	c	time
$T(n) = 2T(n/2) + O(n)$	2	2	1	$O(n \lg n)$
$T(n) = 3T(n/2) + O(n)$	3	2	1	$O(n^{\lg_2 3})$
$T(n) = 3T(n/2) + O(n^2)$	3	2	2	$O(n^2)$

Theorem $T(n) = aT(n/b) + O(n^c)$, where $a \geq 1, b > 1, c \geq 0$ are constants. Then,

$$T(n) = \begin{cases} & \text{if } c < \lg_b a \\ & \text{if } c = \lg_b a \\ & \text{if } c > \lg_b a \end{cases}$$

Master Theorem

Recurrences	a	b	c	time
$T(n) = 2T(n/2) + O(n)$	2	2	1	$O(n \lg n)$
$T(n) = 3T(n/2) + O(n)$	3	2	1	$O(n^{\lg_2 3})$
$T(n) = 3T(n/2) + O(n^2)$	3	2	2	$O(n^2)$

Theorem $T(n) = aT(n/b) + O(n^c)$, where $a \geq 1, b > 1, c \geq 0$ are constants. Then,

$$T(n) = \begin{cases} ?? & \text{if } c < \lg_b a \\ & \text{if } c = \lg_b a \\ & \text{if } c > \lg_b a \end{cases}$$

Master Theorem

Recurrences	a	b	c	time
$T(n) = 2T(n/2) + O(n)$	2	2	1	$O(n \lg n)$
$T(n) = 3T(n/2) + O(n)$	3	2	1	$O(n^{\lg_2 3})$
$T(n) = 3T(n/2) + O(n^2)$	3	2	2	$O(n^2)$

Theorem $T(n) = aT(n/b) + O(n^c)$, where $a \geq 1, b > 1, c \geq 0$ are constants. Then,

$$T(n) = \begin{cases} O(n^{\lg_b a}) & \text{if } c < \lg_b a \\ & \text{if } c = \lg_b a \\ & \text{if } c > \lg_b a \end{cases}$$

Master Theorem

Recurrences	a	b	c	time
$T(n) = 2T(n/2) + O(n)$	2	2	1	$O(n \lg n)$
$T(n) = 3T(n/2) + O(n)$	3	2	1	$O(n^{\lg_2 3})$
$T(n) = 3T(n/2) + O(n^2)$	3	2	2	$O(n^2)$

Theorem $T(n) = aT(n/b) + O(n^c)$, where $a \geq 1, b > 1, c \geq 0$ are constants. Then,

$$T(n) = \begin{cases} O(n^{\lg_b a}) & \text{if } c < \lg_b a \\ & \text{if } c = \lg_b a \\ ?? & \text{if } c > \lg_b a \end{cases}$$

Master Theorem

Recurrences	a	b	c	time
$T(n) = 2T(n/2) + O(n)$	2	2	1	$O(n \lg n)$
$T(n) = 3T(n/2) + O(n)$	3	2	1	$O(n^{\lg_2 3})$
$T(n) = 3T(n/2) + O(n^2)$	3	2	2	$O(n^2)$

Theorem $T(n) = aT(n/b) + O(n^c)$, where $a \geq 1, b > 1, c \geq 0$ are constants. Then,

$$T(n) = \begin{cases} O(n^{\lg_b a}) & \text{if } c < \lg_b a \\ & \text{if } c = \lg_b a \\ O(n^c) & \text{if } c > \lg_b a \end{cases}$$

Master Theorem

Recurrences	a	b	c	time
$T(n) = 2T(n/2) + O(n)$	2	2	1	$O(n \lg n)$
$T(n) = 3T(n/2) + O(n)$	3	2	1	$O(n^{\lg_2 3})$
$T(n) = 3T(n/2) + O(n^2)$	3	2	2	$O(n^2)$

Theorem $T(n) = aT(n/b) + O(n^c)$, where $a \geq 1, b > 1, c \geq 0$ are constants. Then,

$$T(n) = \begin{cases} O(n^{\lg_b a}) & \text{if } c < \lg_b a \\ ?? & \text{if } c = \lg_b a \\ O(n^c) & \text{if } c > \lg_b a \end{cases}$$

Master Theorem

Recurrences	a	b	c	time
$T(n) = 2T(n/2) + O(n)$	2	2	1	$O(n \lg n)$
$T(n) = 3T(n/2) + O(n)$	3	2	1	$O(n^{\lg_2 3})$
$T(n) = 3T(n/2) + O(n^2)$	3	2	2	$O(n^2)$

Theorem $T(n) = aT(n/b) + O(n^c)$, where $a \geq 1, b > 1, c \geq 0$ are constants. Then,

$$T(n) = \begin{cases} O(n^{\lg_b a}) & \text{if } c < \lg_b a \\ O(n^c \lg n) & \text{if } c = \lg_b a \\ O(n^c) & \text{if } c > \lg_b a \end{cases}$$

Theorem $T(n) = aT(n/b) + O(n^c)$, where $a \geq 1, b > 1, c \geq 0$ are constants. Then,

$$T(n) = \begin{cases} O(n^{\lg_b a}) & \text{if } c < \lg_b a \\ O(n^c \lg n) & \text{if } c = \lg_b a \\ O(n^c) & \text{if } c > \lg_b a \end{cases}$$

- Ex: $T(n) = 4T(n/2) + O(n^2)$. Which Case?

Theorem $T(n) = aT(n/b) + O(n^c)$, where $a \geq 1, b > 1, c \geq 0$ are constants. Then,

$$T(n) = \begin{cases} O(n^{\lg_b a}) & \text{if } c < \lg_b a \\ O(n^c \lg n) & \text{if } c = \lg_b a \\ O(n^c) & \text{if } c > \lg_b a \end{cases}$$

- Ex: $T(n) = 4T(n/2) + O(n^2)$. Case 2.

Theorem $T(n) = aT(n/b) + O(n^c)$, where $a \geq 1, b > 1, c \geq 0$ are constants. Then,

$$T(n) = \begin{cases} O(n^{\lg_b a}) & \text{if } c < \lg_b a \\ O(n^c \lg n) & \text{if } c = \lg_b a \\ O(n^c) & \text{if } c > \lg_b a \end{cases}$$

- Ex: $T(n) = 4T(n/2) + O(n^2)$. Case 2. $T(n) = O(n^2 \lg n)$

Theorem $T(n) = aT(n/b) + O(n^c)$, where $a \geq 1, b > 1, c \geq 0$ are constants. Then,

$$T(n) = \begin{cases} O(n^{\lg_b a}) & \text{if } c < \lg_b a \\ O(n^c \lg n) & \text{if } c = \lg_b a \\ O(n^c) & \text{if } c > \lg_b a \end{cases}$$

- Ex: $T(n) = 4T(n/2) + O(n^2)$. Case 2. $T(n) = O(n^2 \lg n)$
- Ex: $T(n) = 3T(n/2) + O(n)$. Which Case?

Theorem $T(n) = aT(n/b) + O(n^c)$, where $a \geq 1, b > 1, c \geq 0$ are constants. Then,

$$T(n) = \begin{cases} O(n^{\lg_b a}) & \text{if } c < \lg_b a \\ O(n^c \lg n) & \text{if } c = \lg_b a \\ O(n^c) & \text{if } c > \lg_b a \end{cases}$$

- Ex: $T(n) = 4T(n/2) + O(n^2)$. Case 2. $T(n) = O(n^2 \lg n)$
- Ex: $T(n) = 3T(n/2) + O(n)$. Case 1.

Theorem $T(n) = aT(n/b) + O(n^c)$, where $a \geq 1, b > 1, c \geq 0$ are constants. Then,

$$T(n) = \begin{cases} O(n^{\lg_b a}) & \text{if } c < \lg_b a \\ O(n^c \lg n) & \text{if } c = \lg_b a \\ O(n^c) & \text{if } c > \lg_b a \end{cases}$$

- Ex: $T(n) = 4T(n/2) + O(n^2)$. Case 2. $T(n) = O(n^2 \lg n)$
- Ex: $T(n) = 3T(n/2) + O(n)$. Case 1. $T(n) = O(n^{\lg_2 3})$

Theorem $T(n) = aT(n/b) + O(n^c)$, where $a \geq 1, b > 1, c \geq 0$ are constants. Then,

$$T(n) = \begin{cases} O(n^{\lg_b a}) & \text{if } c < \lg_b a \\ O(n^c \lg n) & \text{if } c = \lg_b a \\ O(n^c) & \text{if } c > \lg_b a \end{cases}$$

- Ex: $T(n) = 4T(n/2) + O(n^2)$. Case 2. $T(n) = O(n^2 \lg n)$
- Ex: $T(n) = 3T(n/2) + O(n)$. Case 1. $T(n) = O(n^{\lg_2 3})$
- Ex: $T(n) = T(n/2) + O(1)$. Which Case?

Theorem $T(n) = aT(n/b) + O(n^c)$, where $a \geq 1, b > 1, c \geq 0$ are constants. Then,

$$T(n) = \begin{cases} O(n^{\lg_b a}) & \text{if } c < \lg_b a \\ O(n^c \lg n) & \text{if } c = \lg_b a \\ O(n^c) & \text{if } c > \lg_b a \end{cases}$$

- Ex: $T(n) = 4T(n/2) + O(n^2)$. Case 2. $T(n) = O(n^2 \lg n)$
- Ex: $T(n) = 3T(n/2) + O(n)$. Case 1. $T(n) = O(n^{\lg_2 3})$
- Ex: $T(n) = T(n/2) + O(1)$. Case 2.

Theorem $T(n) = aT(n/b) + O(n^c)$, where $a \geq 1, b > 1, c \geq 0$ are constants. Then,

$$T(n) = \begin{cases} O(n^{\lg_b a}) & \text{if } c < \lg_b a \\ O(n^c \lg n) & \text{if } c = \lg_b a \\ O(n^c) & \text{if } c > \lg_b a \end{cases}$$

- Ex: $T(n) = 4T(n/2) + O(n^2)$. Case 2. $T(n) = O(n^2 \lg n)$
- Ex: $T(n) = 3T(n/2) + O(n)$. Case 1. $T(n) = O(n^{\lg_2 3})$
- Ex: $T(n) = T(n/2) + O(1)$. Case 2. $T(n) = O(\lg n)$

Theorem $T(n) = aT(n/b) + O(n^c)$, where $a \geq 1, b > 1, c \geq 0$ are constants. Then,

$$T(n) = \begin{cases} O(n^{\lg_b a}) & \text{if } c < \lg_b a \\ O(n^c \lg n) & \text{if } c = \lg_b a \\ O(n^c) & \text{if } c > \lg_b a \end{cases}$$

- Ex: $T(n) = 4T(n/2) + O(n^2)$. Case 2. $T(n) = O(n^2 \lg n)$
- Ex: $T(n) = 3T(n/2) + O(n)$. Case 1. $T(n) = O(n^{\lg_2 3})$
- Ex: $T(n) = T(n/2) + O(1)$. Case 2. $T(n) = O(\lg n)$
- Ex: $T(n) = 2T(n/2) + O(n^2)$. Which Case?

Theorem $T(n) = aT(n/b) + O(n^c)$, where $a \geq 1, b > 1, c \geq 0$ are constants. Then,

$$T(n) = \begin{cases} O(n^{\lg_b a}) & \text{if } c < \lg_b a \\ O(n^c \lg n) & \text{if } c = \lg_b a \\ O(n^c) & \text{if } c > \lg_b a \end{cases}$$

- Ex: $T(n) = 4T(n/2) + O(n^2)$. Case 2. $T(n) = O(n^2 \lg n)$
- Ex: $T(n) = 3T(n/2) + O(n)$. Case 1. $T(n) = O(n^{\lg_2 3})$
- Ex: $T(n) = T(n/2) + O(1)$. Case 2. $T(n) = O(\lg n)$
- Ex: $T(n) = 2T(n/2) + O(n^2)$. Case 3.

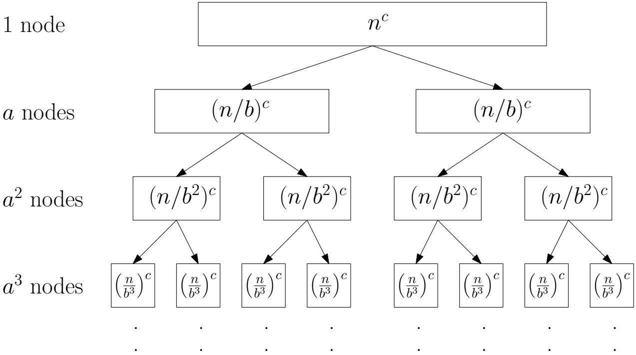
Theorem $T(n) = aT(n/b) + O(n^c)$, where $a \geq 1, b > 1, c \geq 0$ are constants. Then,

$$T(n) = \begin{cases} O(n^{\lg_b a}) & \text{if } c < \lg_b a \\ O(n^c \lg n) & \text{if } c = \lg_b a \\ O(n^c) & \text{if } c > \lg_b a \end{cases}$$

- Ex: $T(n) = 4T(n/2) + O(n^2)$. Case 2. $T(n) = O(n^2 \lg n)$
- Ex: $T(n) = 3T(n/2) + O(n)$. Case 1. $T(n) = O(n^{\lg_2 3})$
- Ex: $T(n) = T(n/2) + O(1)$. Case 2. $T(n) = O(\lg n)$
- Ex: $T(n) = 2T(n/2) + O(n^2)$. Case 3. $T(n) = O(n^2)$

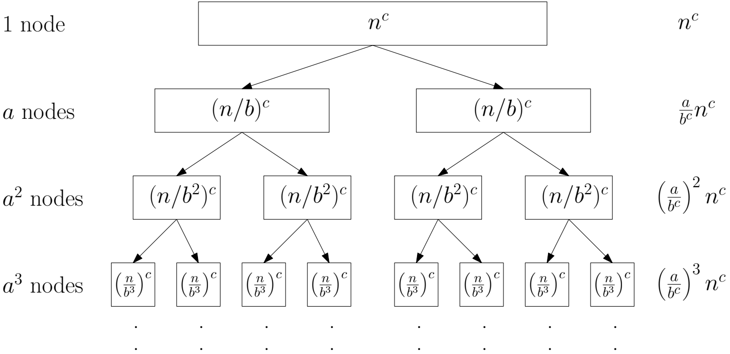
Proof of Master Theorem Using Recursion Tree

$$T(n) = aT(n/b) + O(n^c)$$



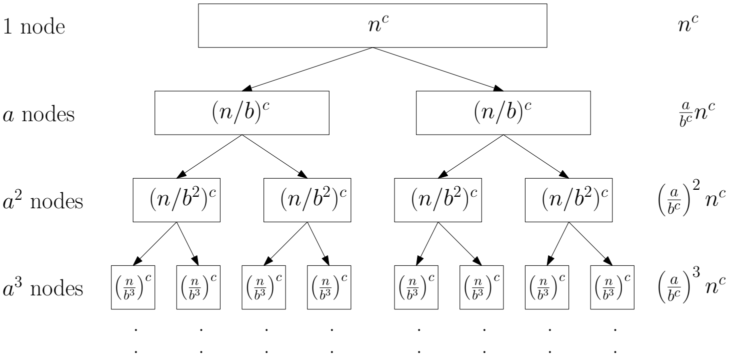
Proof of Master Theorem Using Recursion Tree

$$T(n) = aT(n/b) + O(n^c)$$



Proof of Master Theorem Using Recursion Tree

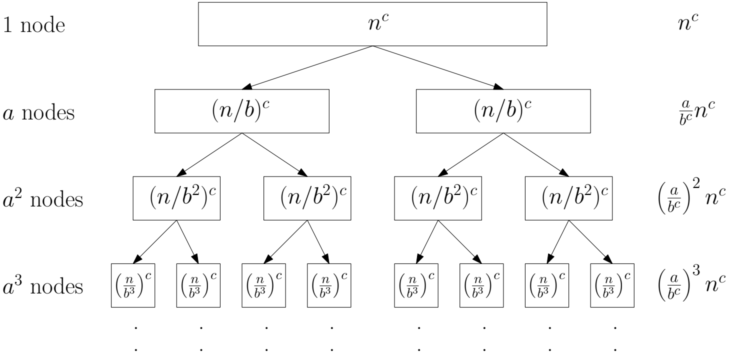
$$T(n) = aT(n/b) + O(n^c)$$



- $c < \lg_b a$: bottom-level dominates: $\left(\frac{a}{b^c}\right)^{\lg_b n} n^c = n^{\lg_b a}$

Proof of Master Theorem Using Recursion Tree

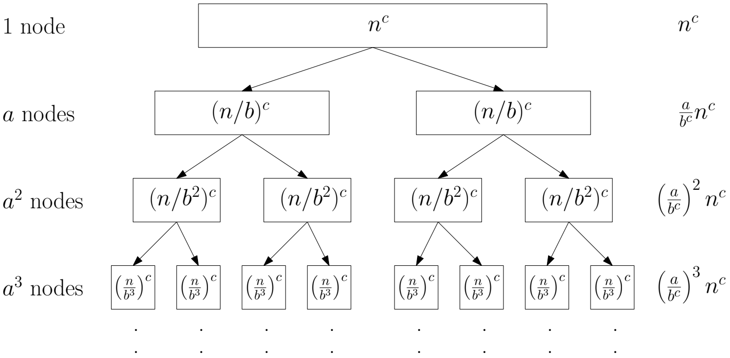
$$T(n) = aT(n/b) + O(n^c)$$



- $c < \lg_b a$: bottom-level dominates: $\left(\frac{a}{b^c}\right)^{\lg_b n} n^c = n^{\lg_b a}$
- $c = \lg_b a$: all levels have same time: $n^c \lg_b n = O(n^c \lg n)$

Proof of Master Theorem Using Recursion Tree

$$T(n) = aT(n/b) + O(n^c)$$



- $c < \lg_b a$: bottom-level dominates: $\left(\frac{a}{b^c}\right)^{\lg_b n} n^c = n^{\lg_b a}$
- $c = \lg_b a$: all levels have same time: $n^c \lg_b n = O(n^c \lg n)$
- $c > \lg_b a$: top-level dominates: $O(n^c)$