

Lecture 3: The Master Theorem; The Recursive Tree Method

1 Quick Review

- Recall, a divide-and-conquer algorithm recursively breaks up a problem of size n in smaller sub-problems such that:
 - There are exactly a sub-problems.
 - Each sub-problem has size at most $\frac{1}{b} \cdot n$
 - Once solved, the solutions to the sub-problems can be combined to produce a solution to the original problem in time $O(n^d)$
- So the run-time of a divide and conquer algorithm satisfies the recurrence:

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + O(n^d)$$

- Examples: MergeSort and Binary Search

2 The Master Theorem

- **The Master Theorem:** If $T(n) = a \cdot T\left(\frac{n}{b}\right) + O(n^d)$ for constants $a > 0$, $b > 1$, and $d \geq 0$ then:

$$T(n) = \begin{cases} O(n^d) & \text{if } a < b^d \text{ [Case I]} \\ O(n^d \cdot \log n) & \text{if } a = b^d \text{ [Case II]} \\ O(n^{\log_b a}) & \text{if } a > b^d \text{ [Case III]} \end{cases}$$

- Sanity Check: What does this give for MergeSort and Binary Search?

| | a | b | d | Case | Runtime |
|---------------|-----|-----|-----|------|---------------------|
| MergeSort | 2 | 2 | 1 | II | $O(n \cdot \log n)$ |
| Binary Search | 1 | 2 | 0 | II | $O(\log n)$ |

3 Proof of The Master Theorem

- Fact One:

Fact 1. $\sum_{k=0}^{\ell} \tau^k = \frac{1 - \tau^{\ell+1}}{1 - \tau}$ for any $\tau \neq 1$.

Proof.

- We have:

$$\begin{aligned} (1 - \tau) \cdot \sum_{k=0}^{\ell} \tau^k &= \sum_{k=0}^{\ell} \tau^k - \sum_{k=1}^{\ell+1} \tau^k \\ &= \tau^0 - \tau^{\ell+1} \\ &= 1 - \tau^{\ell+1} \end{aligned}$$

- Dividing both sides by $(1 - \tau)$ gives the result.

- Fact Two:

Fact 2. $x^{\log_b y} = y^{\log_b x}$ for any base b .

Proof.

$$\log_b z^p = p \cdot \log_b z$$

- Observe that, by the power rule of logarithms, we have:

$$\log_b x \cdot \log_b y = \log_b (y^{\log_b x})$$

- Similarly:

$$\log_b x \cdot \log_b y = \log_b (x^{\log_b y})$$

- Putting this together gives

$$\log_b (y^{\log_b x}) = \log_b (x^{\log_b y})$$

$$\Rightarrow x^{\log_b y} = y^{\log_b x}$$

- Proof of the Master Theorem

Proof.

- We may assume n is a power of b : $n = b^\ell$

- So we have:

$$\begin{aligned} T(n) &= n^d + a \cdot \left(\frac{n}{b}\right)^d + a^2 \cdot \left(\frac{n}{b^2}\right)^d + \cdots + a^\ell \cdot \left(\frac{n}{b^\ell}\right)^d \\ &= n^d \cdot \left(1 + a \cdot \left(\frac{1}{b}\right)^d + a^2 \cdot \left(\frac{1}{b^2}\right)^d + \cdots + a^\ell \cdot \left(\frac{1}{b^\ell}\right)^d\right) \\ &= n^d \cdot \left(1 + \frac{a}{b^d} + \left(\frac{a}{b^d}\right)^2 + \cdots + \left(\frac{a}{b^d}\right)^\ell\right) \end{aligned}$$

Proof [cont.] $T(n) = n^d \cdot \left(1 + \frac{a}{b^d} + \left(\frac{a}{b^d} \right)^2 + \dots + \left(\frac{a}{b^d} \right)^\ell \right)$

Case I: $\frac{a}{b^d} < 1$

- Set $\tau = \frac{a}{b^d}$

- Then: $T(n) = n^d \cdot \sum_{k=0}^{\ell} \tau^k$

- Applying Fact 1, we know that:

$$\sum_{k=0}^{\ell} \tau^k = \frac{1 - \tau^{\ell+1}}{1 - \tau} \leq \frac{1}{1 - \tau} = O(1)$$

As a , b , and d are constants so is $1 - \tau$.

- Therefore:

$$T(n) \leq n^d \cdot \frac{1}{1 - \frac{a}{b^d}} = n^d \cdot \frac{b^d}{b^d - a} = O(n^d)$$

Proof _[cont.] $T(n) = n^d \cdot \left(1 + \frac{a}{b^d} + \left(\frac{a}{b^d} \right)^2 + \cdots + \left(\frac{a}{b^d} \right)^\ell \right)$

Case II: $\frac{a}{b^d} = 1$

- Then: $T(n) = n^d \cdot (\ell + 1)$
- But $n = b^\ell$ so $\ell = \log_b n$.
- As b is a constant greater than one, this gives: $T(n) = O(n^d \cdot \log n)$

Case III: $\frac{a}{b^d} > 1$

- Again set $\tau = \frac{a}{b^d}$
- Then: $T(n) = n^d \cdot \sum_{k=0}^{\ell} \tau^k$
- Applying Fact 1 gives:

$$\sum_{k=0}^{\ell} \tau^k = \frac{\tau^{\ell+1} - 1}{\tau - 1} \leq \frac{\tau^{\ell+1}}{\tau - 1} = O(\tau^{\ell+1}) = O(\tau^\ell)$$

As a , b , and d are constants so is τ .

Proof [cont.]

▪ Thus: $T(n) = O(n^d \cdot \tau^\ell)$

▪ Observe that:

$$n^d \cdot \tau^\ell = n^d \cdot \left(\frac{a}{b^d}\right)^\ell$$

$$= \left(\frac{n}{b^\ell}\right)^d \cdot a^\ell$$

$$= 1 \cdot a^\ell \quad \text{--- As } n=b^\ell.$$

$$= a^{\log_b n}$$

$$= n^{\log_b a} \quad \text{--- By Fact 2.}$$

▪ This gives the final case:

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

- Specifically, what matters is **not** the statement of the Master Theorem but the **ideas** underlying its proof.
 - First, if we understand the proof then we can easily reconstruct the theorem.
 - Second, if we understand the proof then we can easily apply the method to a much broader class of problems. For example:
 - The sub-problems have different sizes.

e.g. The Deterministic Selection Algorithm.

$$T(n) \leq T\left(\frac{7n}{10}\right) + T\left(\frac{n}{5}\right) + O(n)$$

→ The combination function is not of the form $f(n) = n^d$.

e.g. Euclid's Greatest Common Divisor Algorithm.

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

→ The parameters a, b, and d are not constants.

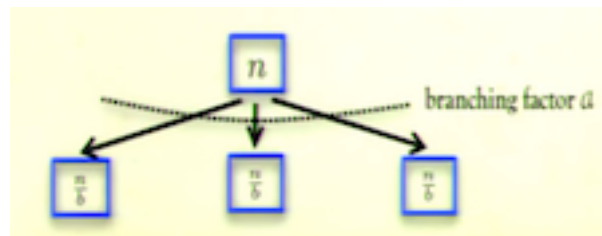
$$\text{e.g. } T(n) = \sqrt{n} \cdot T(\sqrt{n}) + O(n^{\frac{1}{\log \log n}})$$

4 The Recursion Tree Method

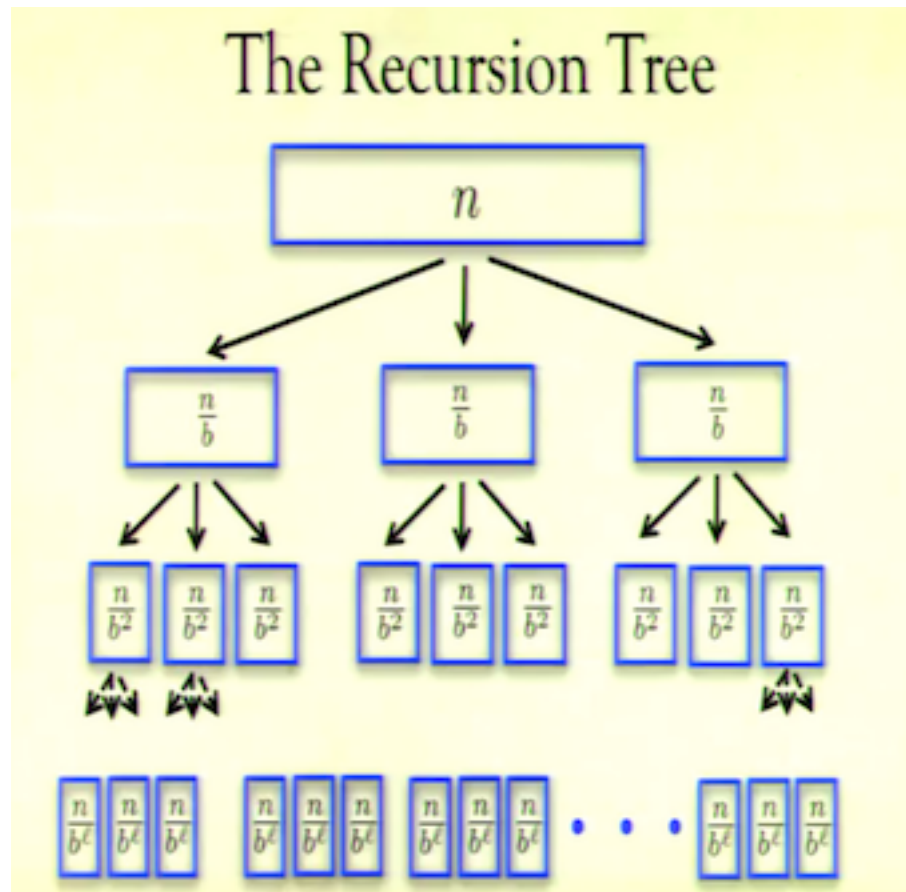
- The Master Theorem is a special case the the **recursion tree method**.
- Specifically, we model the *divide and conquer* recursive formula by a tree:

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + O(n^d)$$

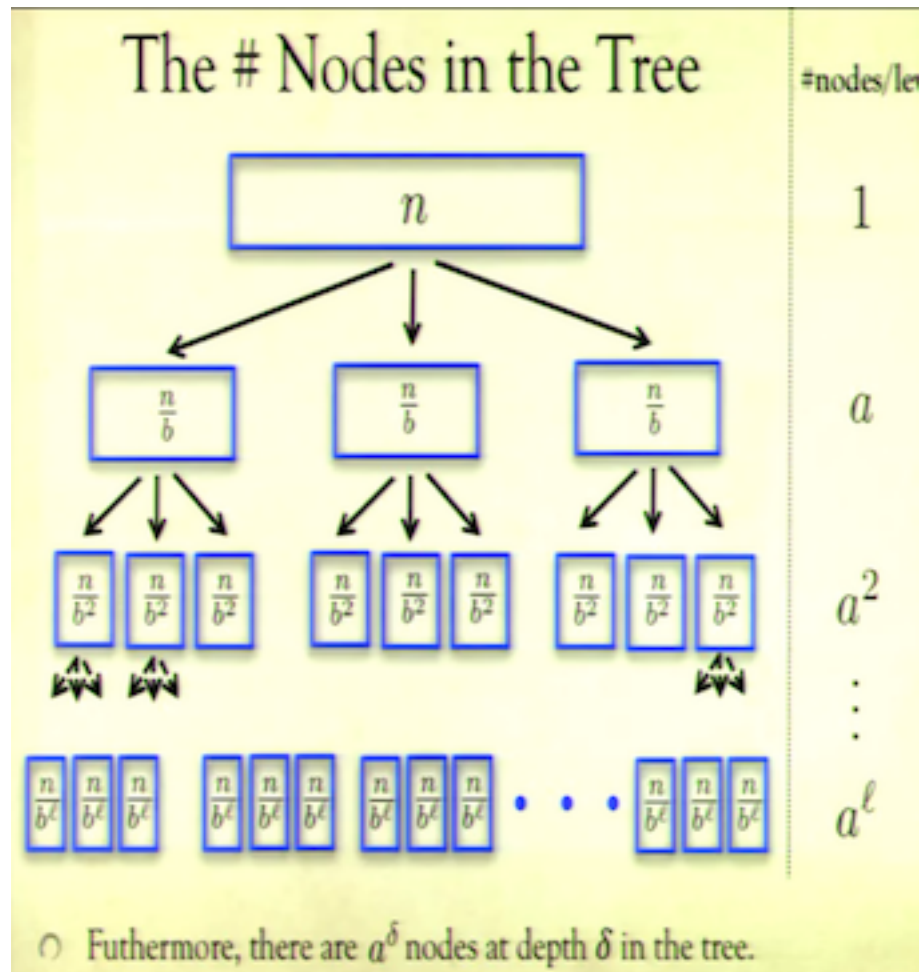
- The **root** node of the trees has a label n.
- The root has a children each with label $\frac{n}{b}$.



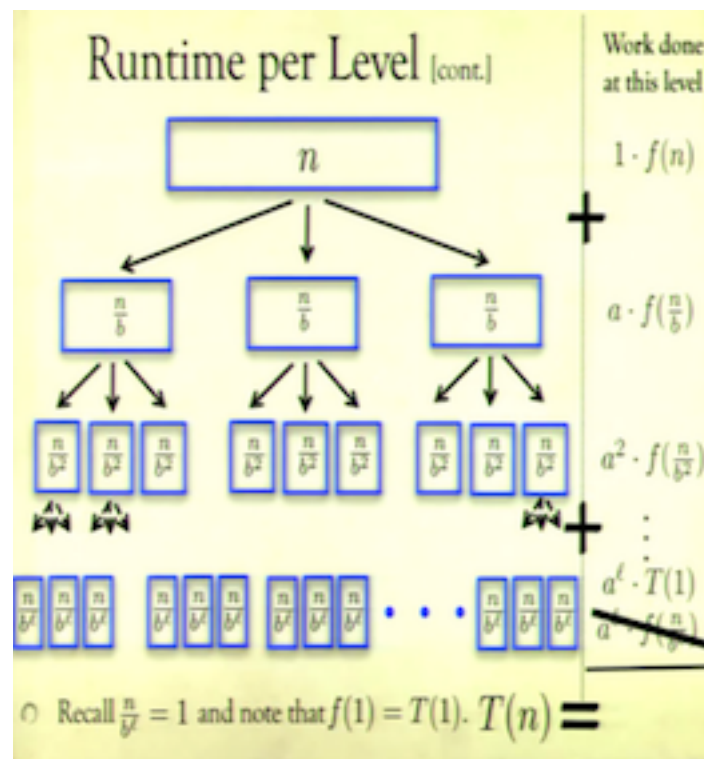
- This pattern then repeats at the children, then grandchildren, etc.

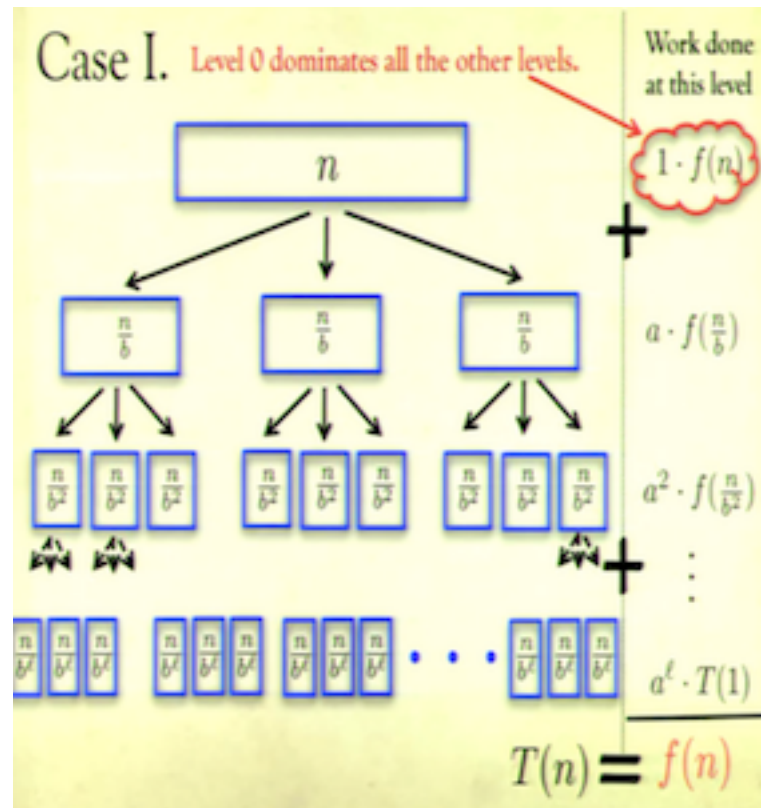


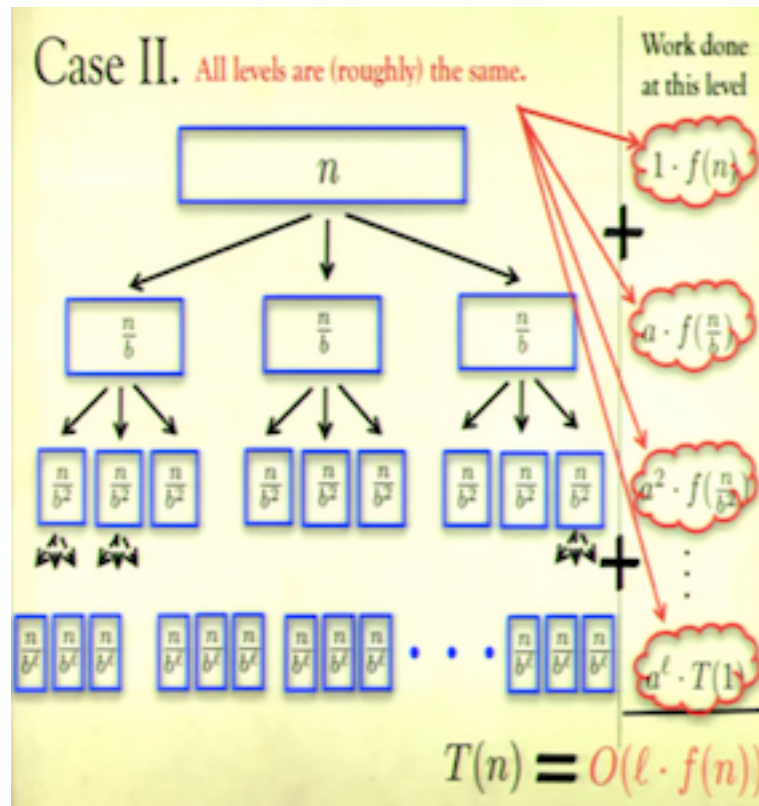
- This process stops at the **leaves** (base cases) which have label $\frac{n}{b^l} = 1$. (As $n = b^l$)

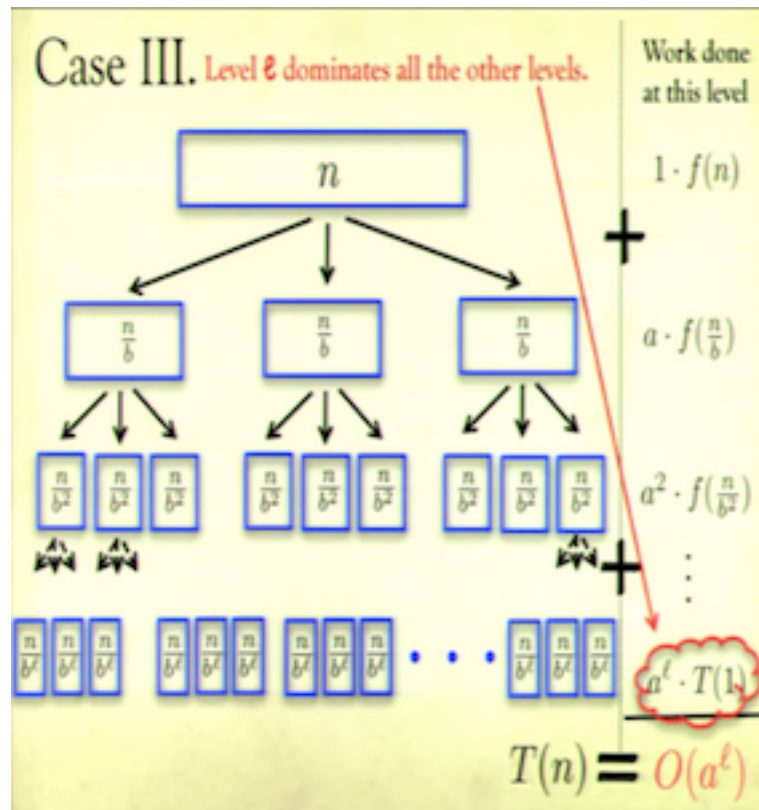


- How much time do we spend at each level?









- This gives us the proof of the Master Theorem:

