Lecture: Analysis of Algorithms (CS583 - 004)

Amarda Shehu

Spring 2019

1. Outline of Today's Class

2. Techniques for Bounding Recurrences
   - Iteration Method
   - Recursion-tree Method
   - Substitution Method
   - Master Theorem

Techniques for Bounding Recurrences

What is a Recurrence?

- A recurrence is an equation of inequality that describes a function in terms of its value on smaller inputs
  - Example: $T(n)$ of Mergesort is described in terms of $T(n/2)$
- Recurrences have boundary conditions (bottom out)
  - Example: $T(n) = c$ when $n = 1$

Techniques for Bounding Recurrences

1. Iteration or expansion method
2. Recursion-tree method
3. Substitution method
4. Master Theorem
5. Generating Functions* (beyond scope of this course)

Iteration Method

Expand $T(n) = 2T(n/2) + cn$ – iterate down to boundary condition

$$\begin{aligned} T(n) &= 2T(n/2) + cn \\ &= 2 \cdot [2T(n/4) + c\tfrac{n}{2}] + cn \\ &= 4 \cdot [2T(n/8) + c\tfrac{n}{4}] + 2cn \\ &= 8 \cdot T(n/8) + 3cn \\ &= 2^3 \cdot T(n/2^3) + 3cn \\ &= \ldots \text{ do you see the pattern?} \\ &= 2^k \cdot T(n/2^k) + kcn \end{aligned}$$

Since the recursion bottoms out at $n = 1$, $k = lg(n)$. So:

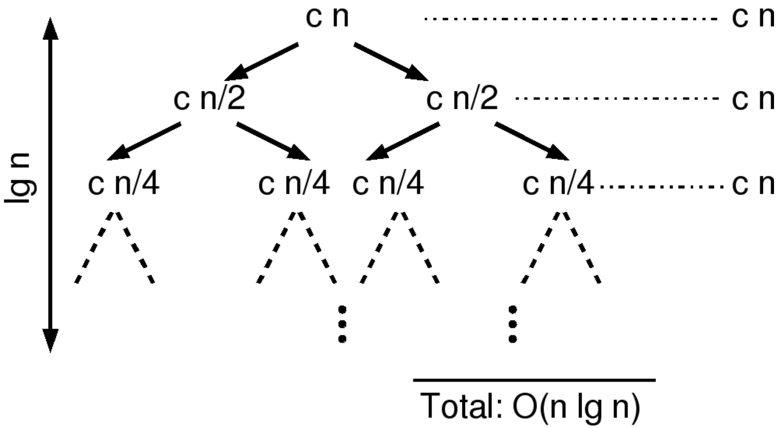
$$\begin{aligned} T(n) &= n \cdot T(1) + lg(n) \cdot cn \\ &= cn + cn \cdot lg(n) \in \theta(n \cdot lgn) \end{aligned}$$

Try to solve $T(n) = T(n-1) + n$, where $T(1) = 1$.

Try to solve $T(n) = 2T(n/2) + n$, where $T(1) = 1$.

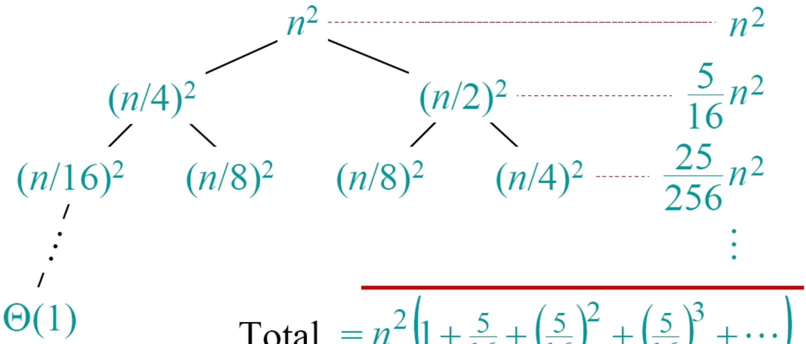
Recursion-tree Method

Build recursion tree for $T(n) = 2T(n/2) + c \cdot n$:



Example of Recursion-tree Method

Solve $T(n) = T(n/4) + T(n/2) + n^2$:



$$\text{Total} = n^2\left(1 + \frac{5}{16} + \left(\frac{5}{16}\right)^2 + \left(\frac{5}{16}\right)^3 + \cdots\right)$$
$$= \Theta(n^2)$$

Substitution (Induction) Method

Guess that $T(n) = 2T(\frac{n}{2}) + n \in O(n + n \cdot lgn)$, where $T(1) = 1$. Then use induction to prove that the guess is correct.

1. **Base Case:** The boundary condition states that $T(1) = 1$. The guess states that $T(1) \in O(1 + 1 \cdot lg1)$. Since, $1 + 1 \cdot lg1 = 1$ and $1 \in O(1)$, the guess is correct.
2. **Inductive Step:** Assuming that $T(\frac{n}{2}) \in O(\frac{n}{2} + \frac{n}{2} \cdot lg(\frac{n}{2}))$, we have to show that the guess holds for $T(n)$:

$$\begin{aligned} T(n) &= 2T(\tfrac{n}{2}) + n \\ &\leq 2[c \cdot (\tfrac{n}{2} + \tfrac{n}{2} \cdot lg(\tfrac{n}{2}))] + n, \text{ where } c > 0 \\ &= c \cdot n + c \cdot n \cdot lgn - cn + n \\ &= c \cdot n \cdot lgn + n \\ & \text{Easy to show that } c \cdot n \cdot lgn + n \in O(n + n \cdot lgn) \end{aligned}$$

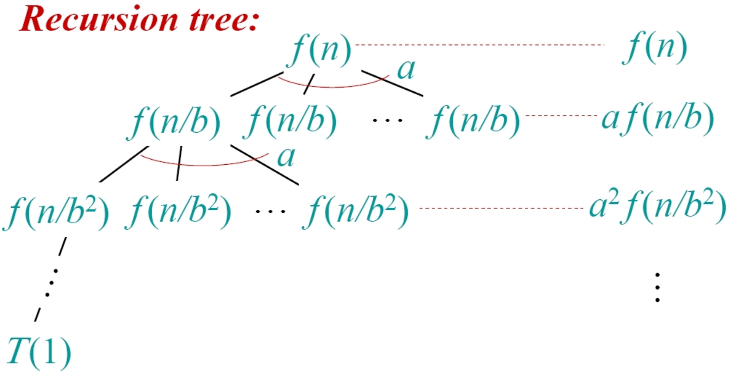
Master Theorem

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 $T(n) = a \cdot T(n/b) + f(n)$, where n/b can mean $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$.

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

Examples: $T(n) = 9T(n/3) + n$, $T(n) = T(\frac{2n}{3}) + 1$, $T(n) = 3T(\frac{n}{4}) + nlgn$, $T(n) = 2T(\frac{n}{2}) + nlgn$, $T(n) = n \cdot T^2(\frac{n}{2})$.

Idea Behind Master Theorem: Case 1.



Idea Behind Master Theorem: Case 1.

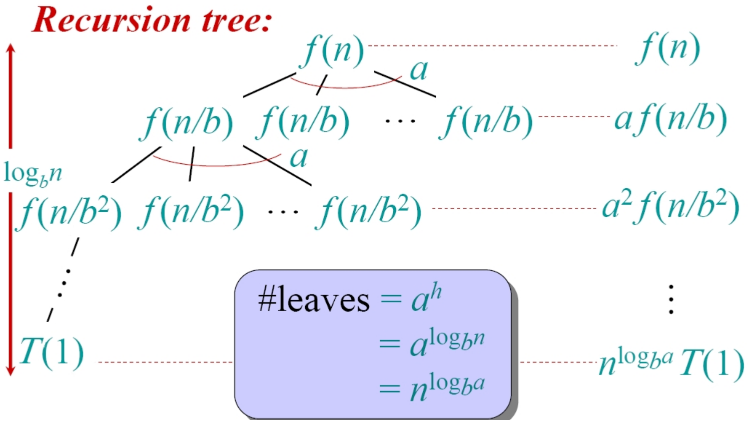
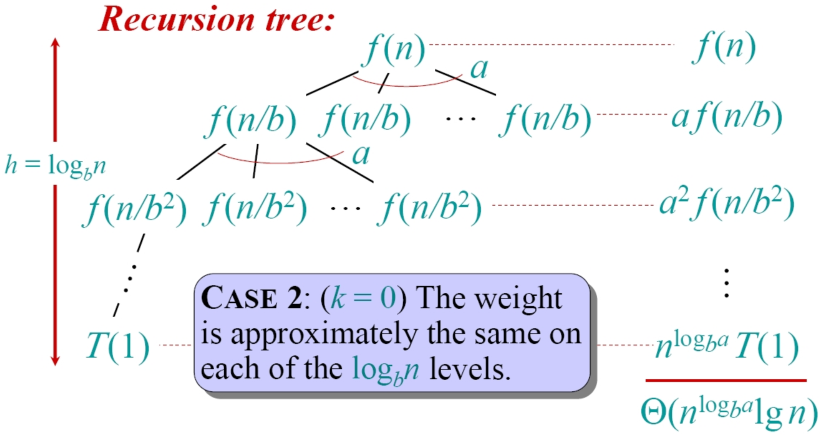


Figure: The weight increases geometrically from the root to the leaves. The leaves hold a constant fraction of the total weight. $T(n) \in \theta(n^{\log_b a})$.

Idea Behind Master Theorem: Case 2.



Idea Behind Master Theorem: Case 3.