Algorithm Design and Analysis

CSE 565

LECTURE 11

Solving Recurrences

- Master Theorem

Sofya Raskhodnikova

Review Question: Exponentiation

Problem: Compute a^b, where $b \in \mathbb{N}$ is n bits long.
Question: How many multiplications?

Naive algorithm: $\Theta(b) = \Theta(2^n)$ (exponential in the input length!)

Divide-and-conquer algorithm:

$$a^b = \begin{cases} a^{b/2} \times a^{b/2} & \text{if } b \text{ is even;} \\ a^{(b-1)/2} \times a^{(b-1)/2} \times a & \text{if } b \text{ is odd.} \end{cases}$$

$T(b) = T(b/2) + \Theta(1) \Rightarrow T(b) = \Theta(\log b) = \Theta(n)$.

So far: 2 recurrences

- Mergesort; Counting Inversions
  $T(n) = 2\,T(n/2) + \Theta(n) \quad = \Theta(n \log n)$
- Binary Search; Exponentiation
  $T(n) = 1\,T(n/2) + \Theta(1) \quad = \Theta(\log n)$

Master Theorem: method for solving recurrences.

The master method

The master method applies to recurrences of the form

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

where $a \geq 1$, $b > 1$, and f is asymptotically positive, that is $f(n) > 0$ for all $n > n_0$.

Three common cases

Compare $f(n)$ with $n^{\log_b a}$:

1. $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$.
   - $f(n)$ grows polynomially slower than $n^{\log_b a}$ (by an n^{ε} factor).

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

Three common cases

Compare $f(n)$ with $n^{\log_b a}$:

1. $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$.
   - $f(n)$ grows polynomially slower than $n^{\log_b a}$ (by an n^{ε} factor).

   Solution: $T(n) = \Theta(n^{\log_b a})$.
2. $f(n) = \Theta(n^{\log_b a} \lg^k n)$ for some constant $k \geq 0$.
   - $f(n)$ and $n^{\log_b a}$ grow at similar rates.

   Solution: $T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$.

Three common cases (cont.)

Compare $f(n)$ with $n^{\log_b a}$:

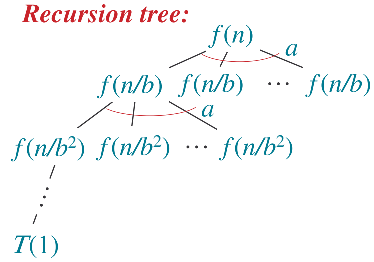
3. $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some constant $\varepsilon > 0$.
   - $f(n)$ grows polynomially faster than $n^{\log_b a}$ (by an n^{ε} factor),

   and $f(n)$ satisfies the ***regularity condition*** that $a f(n/b) \le c f(n)$ for some constant $c < 1$.

   Solution: $T(n) = \Theta(f(n))$.

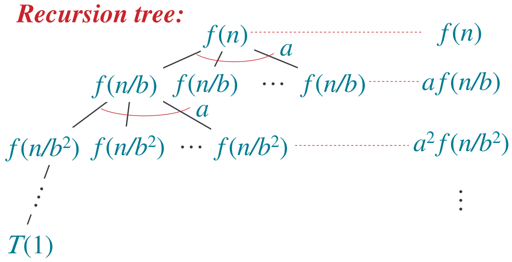
Idea of master theorem

Recursion tree:



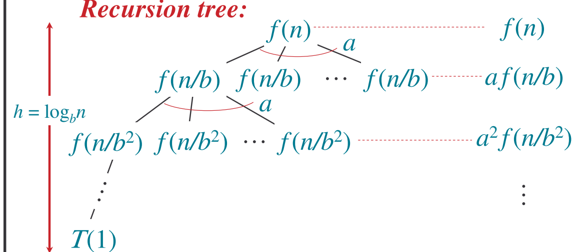
Idea of master theorem

Recursion tree:



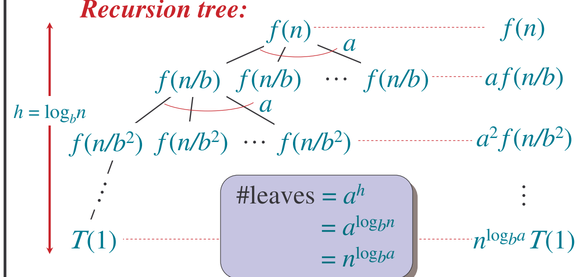
Idea of master theorem

Recursion tree:



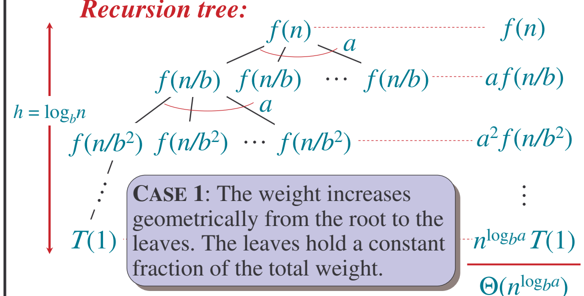
Idea of master theorem

Recursion tree:



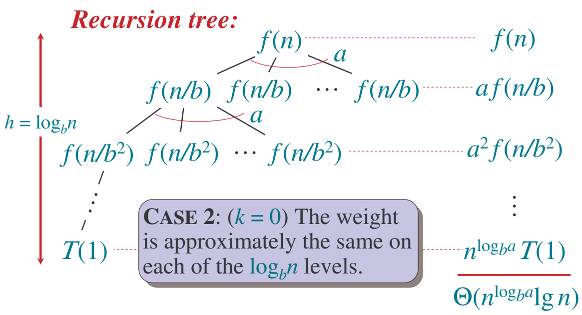
Idea of master theorem

Recursion tree:



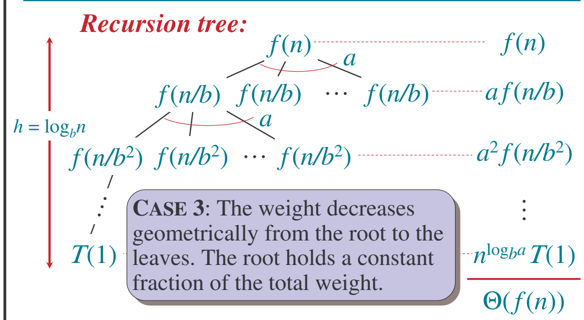
CSE 565

Idea of master theorem



CSE 565

Idea of master theorem



CSE 565

Examples

Ex. $T(n) = 4T(n/2) + n$
$a = 4, b = 2 \Rightarrow n^{\log_b a} = n^2; f(n) = n.$
CASE 1: $f(n) = O(n^{2-\varepsilon})$ for $\varepsilon = 1$.
$\therefore T(n) = \Theta(n^2)$.

CSE 565

Examples

Ex. $T(n) = 4T(n/2) + n$
$a = 4, b = 2 \Rightarrow n^{\log_b a} = n^2; f(n) = n.$
CASE 1: $f(n) = O(n^{2-\varepsilon})$ for $\varepsilon = 1$.
$\therefore T(n) = \Theta(n^2)$.

Ex. $T(n) = 4T(n/2) + n^2$
$a = 4, b = 2 \Rightarrow n^{\log_b a} = n^2; f(n) = n^2.$
CASE 2: $f(n) = \Theta(n^2 \lg^0 n)$, that is, $k = 0$.
$\therefore T(n) = \Theta(n^2 \lg n)$.

CSE 565

Examples

Ex. $T(n) = 4T(n/2) + n^3$
$a = 4, b = 2 \Rightarrow n^{\log_b a} = n^2; f(n) = n^3.$
CASE 3: $f(n) = \Omega(n^{2+\varepsilon})$ for $\varepsilon = 1$
and $4(n/2)^3 \le cn^3$ (reg. cond.) for $c = 1/2$.
$\therefore T(n) = \Theta(n^3)$.

CSE 565

Examples

Ex. $T(n) = 4T(n/2) + n^3$
$a = 4, b = 2 \Rightarrow n^{\log_b a} = n^2; f(n) = n^3.$
CASE 3: $f(n) = \Omega(n^{2+\varepsilon})$ for $\varepsilon = 1$
and $4(n/2)^3 \le cn^3$ (reg. cond.) for $c = 1/2$.
$\therefore T(n) = \Theta(n^3)$.

Ex. $T(n) = 4T(n/2) + n^2/\lg n$
$a = 4, b = 2 \Rightarrow n^{\log_b a} = n^2; f(n) = n^2/\lg n.$
Master method does not apply. In particular, for every constant $\varepsilon > 0$, we have $n^\varepsilon = \omega(\lg n)$.