Master Method

Adapted from the CLRS book slides

MASTER METHOD

Used for many divide-and-conquer ***master recurrences*** of the form $T(n) = aT(n/b) + f(n)$, where a ≥ 1, $b > 1$, and $f(n)$ is an asymptotically nonnegative function defined over all sufficiently large positive numbers.

Master recurrences describe recursive algorithms that divide a problem of size n into a subproblems, each of size n/b. Each recursive subproblem takes time $T(n/b)$ (unless it's a base case). Call $f(n)$ the ***driving function***.

MASTER METHOD (continued)

Based on the ***master theorem*** (Theorem 4.1):

Let $a, b \quad > 0$ be constants, $f(n)$ be a driving function defined and nonnegative on all sufficiently large reals. Define recurrence $T(n)$ on $n \in \mathbb{N}$ by

$$T(n) = aT(n/b) + f(n) ,$$

and where $aT(n/b)$ actually means $a'T(\lfloor n/b \rfloor) + a''T(\lceil n/b \rceil)$ for some constants $a', a'' \geq 0$ satsifying $a = a' + a''$.

MASTER METHOD (continued)

Then you can solve the recurrence by comparing $n^{\log_b a}$ vs. $f(n)$:

Case 1: $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$.
($f(n)$ is polynomially smaller than $n^{\log_b a}$.)
Solution: $T(n) = \Theta(n^{\log_b a})$.
(Intuitively: cost is dominated by leaves.)

Case 2: $f(n) = \Theta(n^{\log_b a} \lg^k n)$, where $k \geq 0$ is a constant.
($f(n)$ is within a polylog factor of $n^{\log_b a}$, but not smaller.)
Solution: $T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$.
(Intuitively: cost is $n^{\log_b a} \lg^k n$ at each level, and there are $\Theta(\lg n)$ levels.)
Simple case: $k = 0 \Rightarrow f(n) = \Theta(n^{\log_b a}) \Rightarrow T(n) = \Theta(n^{\log_b a} \lg n)$.

MASTER METHOD (continued)

Case 3: $f(n) = \Omega(n^{\log_b a+\epsilon})$ for some constant $\epsilon > 0$ and $f(n)$ satisfies the regularity condition $af(n/b) \le cf(n)$ for some constant $c < 1$ and all sufficiently large n.
($f(n)$ is polynomially greater than $n^{\log_b a}$.)
Solution: $T(n) = \Theta(f(n))$.
(Intuitively: cost is dominated by root.)

What's with the Case 3 regularity condition?

- Generally not a problem.
- It always holds whenever $f(n) = n^k$ and $f(n) = \Omega(n^{\log_b a+\epsilon})$ for constant $\epsilon > 0$. So you don't need to check it when $f(n)$ is a polynomial.

MASTER METHOD (continued)

Call $n^{\log_b a}$ the ***watershed function***. Master method compares the driving function $f(n)$ with the watershed function $n^{\log_b a}$.

- If the watershed function grows ***polynomially faster*** than the driving function, then case 1 applies.
- If the driving function grows ***polynomially faster*** than the watershed function and the regularity condition holds, then case 3 applies.
- If the driving function is within a polylog factor of the watershed function but not smaller, then case 2 applies.
- There are gaps between cases 1 and 2 and between cases 2 and 3.

MASTER METHOD EXAMPLES

1

$T(n) = 5T(n/2) + \Theta(n^2)$
$n^{\log_2 5}$ vs. n^2
Since $\log_2 5 - \epsilon = 2$ for some constant $\epsilon > 0$, use case 1 $\Rightarrow T(n) = \Theta(n^{\lg 5})$

2

$T(n) = 27T(n/3) + \Theta(n^3 \lg n)$
$n^{\log_3 27} = n^3$ vs. $n^3 \lg n$
Use case 2 with $k = 1 \Rightarrow T(n) = \Theta(n^3 \lg^2 n)$

3

$T(n) = 5T(n/2) + \Theta(n^3)$
$n^{\log_2 5}$ vs. n^3
Now $\lg 5 + \epsilon = 3$ for some constant $\epsilon > 0$
Check regularity condition (don't really need to since $f(n)$ is a polynomial):
$af(n/b) = 5(n/2)^3 = 5n^3/8 \leq cn^3$ for $c = 5/8 < 1$
Use case 3 $\Rightarrow T(n) = \Theta(n^3)$

4

$T(n) = 27T(n/3) + \Theta(n^3/\lg n)$
$n^{\log_3 27} = n^3$ vs. $n^3/\lg n = n^3 \lg^{-1} n \neq \Theta(n^3 \lg^k n)$ for any $k \geq 0$.
Cannot use the master method.