Recurrence Relations

Recurrence

- A function defined in terms of
  - one or more base cases, and
  - itself, with smaller arguments.

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \,, \\ T(n-1) + 1 & \text{if } n > 1 \,. \end{cases}$$

Solution: $T(n) = n$.

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \,, \\ 2T(n/2) + n & \text{if } n \geq 1 \,. \end{cases}$$

Solution: $T(n) = n \lg n + n$.

Contd...

$$T(n) = \begin{cases} 0 & \text{if } n = 2\,, \\ T(\sqrt{n}) + 1 & \text{if } n > 2\,. \end{cases}$$

Solution: $T(n) = \lg \lg n$.

$$T(n) = \begin{cases} 1 & \text{if } n = 1\,, \\ T(n/3) + T(2n/3) + n & \text{if } n > 1\,. \end{cases}$$

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

Solving Recurrences

- Substitution method,
- Recursion-tree method , and
- Master method.

Master Method

- Provides bounds for recurrences of the form

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

- where $a \geq 1$, $b > 1$, and $f(n)$ is a given function.
- Such recurrences characterizes a divide-and-conquer algorithm that
  - Creates 'a' sub-problems, each of which is '1/b' the size of the original problem.
  - Takes 'f(n)' time in the divide and combine steps together.

Master 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

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

where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ has the following asymptotic bounds:

1. If there exists a constant $\epsilon > 0$ such that $f(n) = O(n^{\log_b a - \epsilon})$, then $T(n) = \Theta(n^{\log_b a})$.
2. If there exists a constant $k \geq 0$ such that $f(n) = \Theta(n^{\log_b a} \lg^k n)$, then $T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$.
3. If there exists a constant $\epsilon > 0$ such that $f(n) = \Omega(n^{\log_b a + \epsilon})$, and if $f(n)$ additionally satisfies the ***regularity condition*** $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$. ■

Compare $n^{\log_b a}$ vs. $f(n)$

Example:

$T(n) = 4T(n/2) + n$

Reading from the equation, $a = 4$, $b = 2$, and $f(n) = n$.

Is $n = O(n^{\log_2 4 - \epsilon}) = O(n^{2-\epsilon})$?

Yes, so case 1 applies and $T(n) = \theta(n^2)$.

Contd…

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

Reading from the equation, $a = 4$, $b = 2$, and $f(n) = n^2$.

Is $n^2 = O(n^{\log_2 4 - \epsilon}) = O(n^{2-\epsilon})$?

No, if $\epsilon > 0$, but it is true if $\epsilon = 0$, so case 2 applies and $T(n) = \Theta(n^2 \log n)$.

Contd…

$T(n) = 4T(n/2) + n^3$

Reading from the equation, $a = 4$, $b = 2$, and $f(n) = n^3$.

Is $n^3 = \Omega(n^{\log_2 4+\epsilon}) = \Omega(n^{2+\epsilon})$?

Yes, for $0 < \epsilon$, so case 3 *might* apply.

Is $4(n/2)^3 \leq c \cdot n^3$?

Yes, for $c \geq 1/2$, so there exists a $c < 1$ to satisfy the regularity condition, so case 3 applies and $T(n) = \Theta(n^3)$.