Recurrences

A *recurrence* is an equation or inequality that describes a function in terms of its value on smaller inputs.

Solving a recurrence means obtaining an asymptotic bound on the solution. There are 3 methods for solving recurrences:

- The **substitution method** where we guess a bound and then prove or disprove it via mathematical induction.
- The **recursion-tree method** which converts the recurrence into a tree whose nodes represent the costs incurred at various levels of the recursion.
- The **master theorem method** which provides bounds for recurrences of the form

  $T(n) = aT(\frac{n}{b}) + f(n)$

  where $a \geq 1, b > 1$ and $f(n)$ is a given function.
  A recurrence of this form characterizes a divide-and-conquer algorithm that creates a subproblems, each of which is $\frac{1}{b}$ the size of the original problem, and in which the divide and combine steps take $f(n)$ time.

The master theorem method

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

$T(n) = aT(\frac{n}{b}) + f(n).$

Then $T(n)$ has the following asymptotic bounds:

1. If $f(n) = O(n^{\log_b a-\epsilon})$ for some constants $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \lg n)$
3. If $f(n) = \Omega(n^{\log_b a+\epsilon})$ for some constants $\epsilon > 0$ and if $af(\frac{n}{b}) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$.