Chapter 4: Recurrences

Recurrences
Recurrence Trees
Master Method

Recurrences

- A recurrence describes a function in terms of its values on smaller inputs.
- The general form of a recurrence is:

$$T(n) = aT(s(n)) + f(n)$$

  where we assume $T(1) \in \Theta(1)$.
- $T(n) = aT(s(n)) + f(n)$ can be interpreted as:
  $a =$ Number of subproblems
  $s(n) =$ Size of the subproblems
  $f(n) =$ Time to divide into subproblems and combine results

Examples of Recurrences

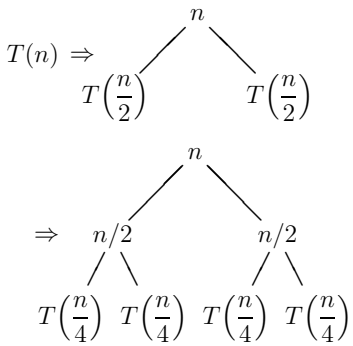
- MERGE-SORT: $T(n) = 2T(n/2) + c\,n$
  2 subproblems
  $n/2 =$ subproblem size
  $\Theta(n)$ time to merge results
- INSERTION-SORT: $T(n) = T(n-1) + c\,n$
  I.e., first sort $n-1$ elts., then insert nth elt.
  1 subproblem
  $n-1 =$ subproblem size
  $\Theta(n)$ time to insert nth element

More Examples of Recurrences

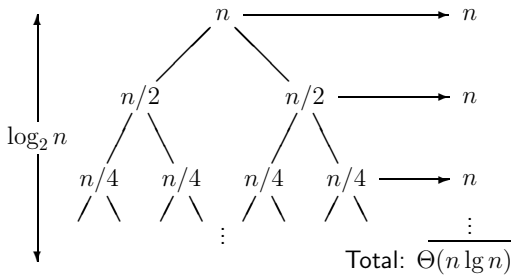
- BIT-MULTIPLY: $T(n) = 3T(n/2) + c\,n$
  3 subproblems (3 multiplications)
  $n/2 =$ subproblem size
  $\Theta(n)$ time to add/sub. results
- STRASSEN'S ALG.: $T(n) = 7T(n/2) + c\,n^2$
  Matrices are $n \times n$
  7 subproblems
  $n/2 \times n/2 =$ subproblem size
  $\Theta(n^2)$ time to form submatrices and add/sub. results

Recursion Trees

Recursion tree for $T(n) = 2T(n/2) + n$



Recursion Tree Continued



Master Method

- For recurrences of the form:

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

- $T(n) \in \Omega(f(n))$ follows immediately.
- $T(n) \in \Omega(n^{\log_b a})$ because the recursion tree has $a^{\log_b n} = n^{\log_b a}$ leaves.
  Height $= \log_b n$.
  Branching factor $= a$.

Master Method Continued

- For recurrences of the form:

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

- $T(n) \in \Theta(f(n))$
  if $f(n)$ is poly. larger than $n^{\log_b a}$ (and is "regular")
- $T(n) \in \Theta(n^{\log_b a})$
  if $f(n)$ is poly. smaller than $n^{\log_b a}$
- $T(n) \in \Theta(n^{\log_b a} \lg n)$
  if $f(n) \in \Theta(n^{\log_b a})$