Chapter 4: Recurrences

Recurrences
Recurrence Trees
Master Method

Recurrences
Recurrence Trees
Master Method

Recurrences
Examples of Recurrences
More Examples of Recurrences
Recursion Trees
Recursion Tree Continued
Master Method
Master Method Continued

Recurrences
Examples of Recurrences

Recurrences

□ A recurrence describes a function in terms of its values on smaller inputs.
□ The general form of a recurrence for running time is:

\[ T(n) = aT(s(n)) + f(n) \]

where we assume \( T(1) \in \Theta(1) \).
□ Interpretation:
\[ a = \text{Number of subproblems} \]
\[ s(n) = \text{Size of the subproblems} \]
\[ f(n) = \text{Time to divide into subproblems and combine results} \]

Examples of Recurrences

□ Merge-Sort:
\[ T(n) = 2T(n/2) + cn \]
2 subproblems of size \( n/2 \)
\( \Theta(n) \) time to merge results

□ Insertion-Sort:
\[ T(n) = T(n - 1) + cn \]
I.e., first sort \( n - 1 \) elts., then insert \( n \)th elt.
1 subproblem of size \( n - 1 \)
\( \Theta(n) \) time to insert \( n \)th element

□ Maximum-Subarray:
\[ T(n) = 2T(n/2) + cn \]
Look at each half, and crossing middle
2 subproblems of size \( n/2 \)
\( \Theta(n) \) time for including middle
More Examples of Recurrences

- **Bit-Multiply**: $T(n) = 3T(n/2) + cn$
  - 3 subproblems (3 multiplications)
  - $n/2 =$ subproblem size
  - $\Theta(n)$ time to add/sub. results

- **Strassen’s Alg.**: $T(n) = 7T(n/2) + cn^2$
  - 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 $T(n) = 2T(n/2) + n$

$$
T(n) \Rightarrow \begin{array}{c}
n \\
T\left(\frac{n}{2}\right) \\
T\left(\frac{n}{2}\right) \\
\Rightarrow \frac{n}{2} \frac{n}{2} \\
T\left(\frac{n}{4}\right) T\left(\frac{n}{4}\right) T\left(\frac{n}{4}\right) T\left(\frac{n}{4}\right) 
\end{array}
$$

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.

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

CS 5633 Analysis of Algorithms
Chapter 4: Slide – 4

CS 5633 Analysis of Algorithms
Chapter 4: Slide – 5

CS 5633 Analysis of Algorithms
Chapter 4: Slide – 6

CS 5633 Analysis of Algorithms
Chapter 4: Slide – 7
Master Method Continued

- For recurrences of the form:

  \[ T(n) = aT(n/b) + f(n) \]

- Let \( c = \log_b a \)

- \( T(n) \in \Theta(n^c) \)
  - if \( f(n) \) is \( O(n^d) \) and \( d < c \)

- \( T(n) \in \Theta(n^c \lg n) \)
  - if \( f(n) \in \Theta(n^c) \)

- \( T(n) \in \Theta(f(n)) \)
  - if \( f(n) \) is \( \Omega(n^d) \) and \( d > c \) (and \( f(n) \) is “regular”)