Introduction to Computers and Programming

Prof. I. K. Lundqvist

Reading:

Lecture 13
Oct 2 2003
Recap

• Iteration versus Recursion

• Towers of Hanoi

• Computed time taken to solve towers of Hanoi
Divide and Conquer

- It is an algorithmic design paradigm that contains the following steps
  - **Divide**: Break the problem into smaller sub-problems
  - **Recur**: Solve each of the sub-problems recursively
  - **Conquer**: Combine the solutions of each of the sub-problems to form the solution of the problem

Represent the solution using a recurrence equation
Recurrence Equation

• A recurrence equation is of the form
  \[ T(n) = aT(m) + b, \quad n > 0, \ m < n \]
  (induction)
and
  \[ T(0) = \text{constant} \]
  (base case)

Where:
  - \( aT(m) \): cost of solving a sub-problems of size \( m \)
  - \( b \): cost of pulling together the solutions
Solving Recurrence Equations

• Iteration
• Recurrence Trees
• Substitution
• Master Method
Towers of Hanoi

Given: \( T(1) = 1 \)

\[ T(n) = 2 \times T(n-1) + 1 \]

<table>
<thead>
<tr>
<th>N</th>
<th>No. Moves</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>15</td>
</tr>
<tr>
<td>5</td>
<td>31</td>
</tr>
</tbody>
</table>
Using Iteration

\[ T(n) = 2 \cdot T(n-1) + 1 \]
\[ T(n) = 2 \left[ 2 \cdot T(n-2) + 1 \right] + 1 \]
\[ T(n) = 2 \left[ 2 \left[ 2 \cdot T(n-3) + 1 \right] + 1 \right] + 1 \]
\[ T(n) = 2 \left[ 2 \left[ 2 \left[ 2 \cdot T(n-4) + 1 \right] + 1 \right] + 1 \right] + 1 \]
\[ T(n) = 2^4 \cdot T(n-4) + 15 \]

... 

\[ T(n) = 2^k \cdot T(n-k) + 2^k - 1 \]

Since \( n \) is finite, \( k \to n \).
Therefore,

\[ \lim_{k \to n} T(n) = 2^n - 1 \]
Greatest Common Divisor

Given two natural numbers $a, b$
- If $b = 0$, then $\text{GCD} := a$
- If $b \neq 0$, then
  - $c := a \mod b$
  - $a := b$
  - $b := c$
  - $\text{GCD}(a,b)$
[The MOD function]

• Notation: \( m \mod n = x \)

• \( x \) = integer remainder when \( m \) is divided by \( n \)
  \[ = m - \left\lfloor \frac{m}{n} \right\rfloor n \]

• Examples:
  - 8 mod 3 = 2
  - 42 mod 6 = 0
  - 5 mod 7 = 5
Extended Euclid’s Algorithm

\[ \text{GCD}(a,b) = ap + bq \]

\[ \begin{align*}
38 \mod 10 &= 8 \\
10 \mod 8 &= 2 \\
8 \mod 2 &= 0 \\
\text{GCD}(2,0) &= 2
\end{align*} \]

\[ 38 \mod 10 = 8 = 38 - 3 \times 10 \]
\[ 10 \mod 8 = 2 = 10 - 1 \times 8 \]
\[ 8 \mod 2 = 0 = 10 - 1 \times (38 - 3 \times 10) \]
\[ = 4 \times 10 - 1 \times 38 \]

“2” can be expressed as linear combination of 10 and 38 – Solve Diophantine Equations
Exercise

• Write 6 as an integer combination of 10 and 38
  – Find GCD (38,10)
  – Express the GCD as a linear combination of 38 and 10
  – Multiply that expression by (6/GCD)

\[ 6 = 3 \left( 4 \times 10 - 1 \times 38 \right) \]
\[ = 12 \times 10 - 3 \times 38 \]
Multiplication

• Standard method for multiplying long numbers:
  \[(1000a+b) \times (1000c+d) = 1,000,000 ac + 1000 (ad + bc) + bd\]

• Instead use:
  \[(1000a+b) \times (1000c+d) = 1,000,000 ac + 1000 ((a+b)(c+d) - ac - bd) + bd\]

One length-k multiply = 3 length-k/2 multiplies and a bunch of additions and shifting
[Logarithms – $\log_b(x)$]

- A logarithm of base $b$ for value $y$ is the power to which $b$ is raised to get $y$.
  - $\log_b y = x \iff b^x = y \iff b^{\log_b y} = y$
  - $\log_b 1 = 0$, $\log_b b = 1$ for all values of $b$
Given \( n, \ n \log n, \ n^2, \ n(\log n)^2 \), for large \( n \):

1. \( n \) has the largest value
2. \( n \log n \) has the largest value
3. \( n^2 \) has the largest value
4. \( n(\log n)^2 \) has the largest value
Relative size of $n$, $n \log n$, $n^2$, $n(\log n)^2$

- $(n \log n)/n = \log n \to \infty$
  $n$ is more efficient than $\log n$

- $n(\log n)^2 / n \log n = \log n \to \infty$
  $n \log n$ is more efficient than $n(\log n)^2$

- $n(\log n)^2 / n^2 = (\log n)^2 / n \to 0$
  $n(\log n)^2$ is more efficient than $n^2$

- Order of efficiency is $n$, $n \log n$, $n(\log n)^2$, $n^2$
Recurrence Tree

Recurrence Equation: \( T(n) < 3T(n/2) + cn \)

- 1 node at depth-0
- 3 nodes at depth-1
- 9 nodes at depth-2
- \( 3^{\log n} \) nodes at depth-\( \log n \)
Solving using Recurrence Tree

\[ T(n) < cn \left( 1 + 3^{1/2} + 9^{1/4} + \ldots + 3^{\log n} \left( \frac{1}{2^{\log n}} \right) \right) \]
\[ < cn \left( 1 + \frac{3}{2} + \left( \frac{3}{2} \right)^2 + \ldots + \left( \frac{3}{2} \right)^{\log n} \right). \]

\[ < cn \left( \left( \frac{3}{2} \right)^{\log n + 1} - 1 \right) / ((3/2)-1) \]
\[ < cn \left( \left( \frac{3}{2} \right)^{\log n} \left( \frac{3}{2} \right) - 1 \right) / (1/2) \]
\[ < \left( \left( cn \left( \frac{3}{2} \right)^{\log n} \left( \frac{3}{2} \right) \right)/(1/2) \right)-2cn \]
\[ < cn \left( \frac{3}{2} \right)^{\log n} - 2cn. \]

\[ T(n) < 3 \left( \frac{3}{2} \right)^{\log n} \quad --\text{approximation} \]

\[ 3cn \left( n^{\log(3/2)} \right) = 3c \left( n^{1+\log(3/2)} \right) \]
Important Theorems

Arithmetic Series
For \( n \geq 1 \), \( 1 + 2 + \ldots + n = \frac{n(n+1)}{2} \)

Geometric Series
For \( a \geq 1 \), \( a^k + a^{k-1} + \ldots + 1 = \frac{a^{k+1} - 1}{a-1} \)

Logarithmic Behavior
\( a \ lg \ b = b \ lg \ a \)
Recurrence Examples

\[
s(n) = \begin{cases} 
  0 & n = 0 \\
  c + s(n-1) & n > 0
\end{cases}
\]

\[
s(n) = \begin{cases} 
  0 & n = 0 \\
  n + s(n-1) & n > 0
\end{cases}
\]

\[
T(n) = \begin{cases} 
  c & n = 1 \\
  2T\left(\frac{n}{2}\right) + c & n > 1
\end{cases}
\]

\[
T(n) = \begin{cases} 
  c & n = 1 \\
  aT\left(\frac{n}{b}\right) + cn & n > 1
\end{cases}
\]