More about Recurrence Relations

Four solution techniques

Recap: A useful mathematical fact

$$x^{\log_b y} = y^{\log_b x}$$

Can you see why this is true for all positive values of x, y, and b?

Recurrence relation basics

- What does the solution of a recurrence relation describe?
  - a (usually infinite) sequence
  - how is this like solving a D.E.?
- What is the form of a recurrence relation?
  1. initial value(s): value for c_0, or perhaps c_0, c_1, …c_k for some fixed number k
  2. a general formula that relates c_n to one or more earlier values in the sequence

Solving recurrence relations Technique 1: ad hoc methods

- Recognize a pattern.
  - That's pretty much what we did in the previous examples.

Another Example: Write and analyze selection sort

- Code
  - `void sort (int [ ] a)`
- Analysis:
  - Write a recurrence relation
  - solve it

Technique 2: iteration

- Example: $f_1 = 1$, $f_n = n + f_{n/2}$, where n is a power of 2 (i.e., $n = 2^k$ for some k)
- Iterate some terms to see the pattern
- What if n is not a power of 2?
  - It is still true that f_n is $\Theta(n)$
  - Details are too complex for this course
  - Take CSSE 473

Interlude

The GOTO statement is the seed from which all other iteration statements have been germinated

Unfortunately, it is a semolina seed, producer of spaghetti code and endless confusion

-- Jesse Liberty: Programming C#,
2nd edition, page 43

Technique 3: telescoping

- From the recursive max contiguous subsequence sum algorithm:
- $f_1 = 1$, $f_n = 2f_{n/2} + n$, where n is a power of 2. ($n = 2^k$)
- Divide both sides by n to get

$$\frac{f_n}{n} = \frac{f_{n/2}}{n/2} + 1$$

- Substitute n/2, n/4, …, 2 for n to get several equations.
- Add the left and right sides of the equations
- Notice the terms that cancel
- Simplify and plug in the initial value
- Result: $f_n = n \log n + n$

Technique 4: catalog some general equations and their solutions

- General Divide and Conquer Example:
- $f_n = Af_{n/B} + h(n)$, where $h(n)$ is $O(n^k)$, and where $A \geq 1$ and $B > 1$
- Solution:

$$f_n = \begin{cases} O\left(n^{\log_B A}\right) & if \quad A > B^k \\ O\left(n^k \log n\right) & if \quad A = B^k \\ O\left(n^k\right) & if \quad A < B^k \end{cases}$$

Soon we will show that this is true

Famous Diversion – Towers of Hanoi (a relevant interlude)

- The Towers of Hanoi puzzle was invented by the French mathematician Edouard Lucas in 1883.
- We are given a tower of disks initially stacked in decreasing size on one of three pegs
- The objective is to transfer the entire tower to one of the other pegs,
- moving only one disk at a time and
- never placing a larger disk on top of a smaller disk

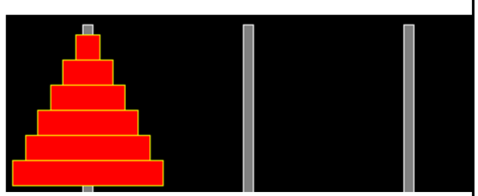

Image is from
http://www.cut-the-knot.com/recurrence/hanoi.html

Towers of Hanoi – (my)hands on

Demo!

Towers of Hanoi

- Write the method (and its recursive helper)
- Analyze it: count the total moves required to move n disks from one peg to another
  - I.e., write and solve the recurrence relation

Technique 4: catalog some general equations and their solutions

- General Divide and Conquer Example:
- $f_n = Af_{n/B} + h(n)$, where $h(n)$ is $O(n^k)$, and where $A \geq 1$ and $B > 1$
- Solution:

$$f_n = \begin{cases} O\left(n^{\log_B A}\right) & if \quad A > B^k \\ O\left(n^k \log n\right) & if \quad A = B^k \\ O\left(n^k\right) & if \quad A < B^k \end{cases}$$

Soon we will show that this is true

Why will we do this long derivation of this solution?

- We'll use the result a few times
- Helps you understand a technique (telescoping) that works for solving many other recurrence relations
- Some of the detailed steps are the kinds of things you'll be doing again, and repetition does not hurt!

Proof of a special case of this general divide-and-conquer recurrence relation

- $f_n = Af_{n/B} + h(n)$, where $h(n)$ is $O(n^k)$, $A \geq 1$ and $B > 1$.
- Solution:

$$\boxed{f_n = \begin{cases} O\left(n^{\log_B A}\right) & \text{if} \quad A > B^k \\ O\left(n^k \log n\right) & \text{if} \quad A = B^k \\ O\left(n^k\right) & \text{if} \quad A < B^k \end{cases}}$$

- Special case:
  - Assume that n is a power of B ($n=B^M$), that $f_1=1$, and ignore the constant factor in $O(n^k)$: [i.e., use $h(n) = n^k$]
- The recurrence relation becomes

$$f_{B^M} = A \; f_{B^{M-1}} + B^{kM}$$