Recurrence Relations

Recurrence Relations & Recursion

Computer Science has recursion

Mathematics has recurrence relations.

Example:

$s_n = s_{n-1} - 3, \;\; s_1 = 13, \;\forall n \in \mathbb{Z}$ where $1 \leq n \leq 5$ defines the sequence $13, 10, 7, 4, 1$

The Fibonacci sequence is defined by the recurrence

$$f_n = f_{n-1} + f_{n-2}$$

Where $f_0 = 0$ and $f_1 = 1$

Recurrence Relations

Definition: *Recurrence Relation*

A recurrence relation for the sequence $a_0, a_1, \ldots$ is an equation that expresses a_k in terms of one or more of its preceding sequence members, one or more of which are initial conditions for the sequence

Example:

The number of subsets of a set of n elements:

$s(0) = 1$ is the *initial condition*

$s(n) = 2 \cdot s(n-1)$ is the *recurrence relation*

Recall: This is the cardinality of a power set.

Solving Recurrence Relations

Given a recurrence relation, can an equivalent closed-form (non-recurrence) expression (a.ka. an explicit formula) be produced?

If so, the closed-form expression is the *solution* to the recurrence relation

Utility: Solving recurrence relations is a common task is <u>algorithm analysis</u>

Linear Homogeneous Recurrence Relations

Definition: *Linear Homogeneous Recurrence Relation With Constant Coefficients (LHRRWCC) of Degree k*

A LHRRWCC of degree k has the form:

$R(n) = c_1 R(n-1) + c_2 R(n-2) + \cdots + c_k R(n-k)$

where $c_i \in \mathbb{R}$ and $c_k \neq 0$

Example:

$S(n) = 2 \cdot S(n-1)$ is a LHRRWCC of degree 1

$f_n = f_{n-1} + f_{n-2}$ is a LHRRWCC of degree 2

$A(x) = A(x-2)$ is alos a LHRRWCC of degree 2

Solving LHRRWCCs of Degree 2

Assumption: $R(n) = w^n$ where w is a non-zero constant.
(Why? We'll get a nice quadratic expression at the end!)

If $R(n) = w^n$, then $R(n-1) = w^{n-1}$, etc.

Thus: $R(n) = c_1R(n-1) + c_2R(n-2) + \cdots + c_kR(n-k)$

becomes: $w_n = c_1w^{n-1} + c_2w^{n-2} + \cdots + c_kw^{n-k}$

As our degree is 2, we need only terms $k = 1$ and $k = 2$:
$w^n = c_1w^{n-1} + c_2w^{n-2}$

Divide through by $w^{n-2} \Rightarrow w^2 = c_1w^1 + c_2$

Rearrange: $\Rightarrow w^2 - c_1w^1 - c_2 = 0$

Solving LHRRWCCs of Degree 2

Theorem: Assume a characteristic equation $w^2 - c_1 w - c_2 = 0$ with $c_1, c_2 \in \mathbb{R}$ and roots r_1 and r_2 such that $r_1 \neq r_2$. The sequence $\{R(n)\}$ is a solution to $R(n) = c_1 R(n-1) + c_2 R(n-2)$ iff $R(n) = \alpha_1 r_1^n + \alpha_2 r_2^n$ where $n \in \mathbb{Z}^*$ and $\alpha_a, \alpha_2 \in \mathbb{R}$.

Proof: Rosen Sect. 8.2 p 542-3

Solution Procedure: LHRRWCCs of Degree 2

1. Identify c_1 & c_2 and create the characteristic equation $w^2 - c_1 w - c_2 = 0$

2. Insert the roots of the characteristic equation (r_1 & r_2) into the closed-form expression $R(n) = \alpha_1 r_1^n + \alpha_2 r_2^n$

3. Using the initial conditions, create two equations in two unknowns (α_1 and α_2)

4. Solve for α_1 and α_2 to complete the solution

Example: Solving a LHRRWCC of Degree 2

Solve: $R(n) = 3R(n-1) - 2R(n-2)$

where $R(0) = 200$ and $R(1) = 220$

(1) From the recurrence, we see that $c_1 = 3$ and $c_2 = -2$
$\therefore$ Characteristic eq. Is $w^2 - 3w - (-2) = w^2 - 3w + 2 = 0$

(2) Factor: $w^2 - 3w + 2 = (w-2)(w-1)$.
It follows that the roots are: $r_1 = 2$ and $r_2 = 1$.
And so: $R(n) = \alpha_1 2^n + \alpha_2 1^n = \alpha_1 2^n + \alpha_2$

(3) Apply the initial conditions to $R(n) = \alpha_1 r_1^n + \alpha_2 r_2^n$:
$R(0) = \alpha_1 + \alpha_2 = 200$ $\qquad$ $R(1) = 2\alpha_1 + \alpha_2 = 220$

(4) Solve for the two unknowns: $\alpha_1 = 20$ and $\alpha_2 = 180$.

Thus the solution is $R(n) = 20 \cdot 2^n + 180 \cdot 1^n = 20 \cdot 2^n + 180$

“Divide & Conquer” Recurrence Relations

- Background:
  - “Divide and Conquer” is a military, political, and algorithmic tactic:
  - Military: Disconnect one half of a battle group from the other, and the two halves are much easer to defeat
  - Political: Force the liberal and conservative wings of a political party to squabble, and the other party finds its work to be more easily accomplished
  - Algorithmic: Solving two halves of a problem (and combining the results to construct the original problem’s answer) is often more efficient than solving the original problem directly

"Divide & Conquer" Recurrence Relations

Example:

(1) Binary Search

$$S(1) = 1$$
$$S(n) = S(\frac{n}{2}) + k$$

(2) Best Case of Quicksort

$$Q(1) = 1$$
$$Q(n) = \quad Q(\frac{n}{2}) \quad + \quad Q(\frac{n}{2}) \quad + \quad n$$

[Worst case of Quicksort: $Q(n) = Q(n-1) + n$]

Solving "Divide & Conquer" Rec. Relations

"Find The Pattern" (a.k.a. Iterative (or Backward) Substitutions)

Example:

$$S(1) = 1$$

$$S(n) = S(\frac{n}{2}) + k$$

$$S(\frac{n}{2}) = S(\frac{n}{4}) + k \quad \Rightarrow \quad S(n) = S(\frac{n}{4}) + 2k$$

$$S(\frac{n}{4}) = S(\frac{n}{8}) + k \quad \Rightarrow \quad S(n) = S(\frac{n}{8}) + 3k$$

$$S(\frac{n}{8}) = S(\frac{n}{16}) + k \quad \Rightarrow \quad S(n) = S(\frac{n}{16}) + 4k$$

(continues…)

Solving “Divide & Conquer” Rec. Relations

In general: $S(n) = S(\frac{n}{2^a}) + ak$, where $a \geq 1, \quad a \in \mathbb{Z}$

[Simplifying assumption: n is a power of 2]

Let $n = 2^a$; that is, $a = \log_2 n$

$S(n) = S(\frac{n}{n}) + k \log_2 n$

$S(n) = S(1) + k \log_2 n$

$S(n) = 1 + k \log_2 n$

$\therefore S(n)$ is $O(\log_2 n)$

But … do you believe?

Solving "Divide & Conquer" Rec. Relations

Conjecture: $S(n) = k \cdot \log_2 n + 1$

Proof (weak induction):

Basis: $n = 1.$ $S(1) = 1 = k \cdot 0 + 1 = k \cdot \log_2 1 + 1$

Inductive Step: If $S(j) = k \cdot \log_2 j + 1$ then $S(2j) = k \cdot \log_2(2j) + 1$

$S(2j) = S(\frac{2j}{2}) + k$ **Applying the Recurrence**

$= S(j) + k$ **Simplifying**

$= k \cdot \log_2 j + 1 + k$ **By the Inductive Hypothesis**

$= k(\log_2 j + 1) + 1$

$= k(\log_2 j + \log_2 2) + 1$

$= k \cdot \log_2(2j) + 1$ **As we needed to show**

Therefore, $S(n) = k \cdot \log_2 n + 1$

Solving "Divide & Conquer" Rec. Relations

Example: Worst Case of Quicksort

Recall: $Q(1) = 1$, and $Q(n) = Q(n-1) + n$

$$Q(n) = Q(n-1) + n$$

$$Q(n-1) = Q(n-2) + (n-1)$$

$$Q(n) = Q(n-2) + n + (n-1)$$

$$Q(n-2) = Q(n-3) + (n-2)$$

$$Q(n) = Q(n-3) + n + (n-1) + (n-2)$$

Apparently, in general:

$$Q(n) = Q(n-k) + \sum_{i=0}^{k-1} (n-i), k \in \mathbb{Z}^+$$

(continues…)

Solving "Divide & Conquer" Rec. Relations

If we continue, we'll reach $Q(n-k) = Q(1)$ when $k = n-1$

$$Q(n) = Q(n-(n-1)) + \sum_{i=1}^{(n-1)-1} (n-i)$$ **Substituting**

$$= Q(1) + \sum_{i=2}^{n} i$$ **Simplifying**

$$= 1 + \sum_{i=2}^{n} i$$ **Combine Terms**

$$= \frac{n(n+1)}{2}$$ **By Gauss**

And this shows why Quicksort is $O(n^2)$ in the worst case…

… But do you believe?

Solving "Divide & Conquer" Rec. Relations

Conjecture: $Q(n) = \dfrac{n(n+1)}{2}$

Proof (weak induction):

Basis: $n = 1.\;\; Q(1) = 1 = \dfrac{2}{2} = \dfrac{1(1+1)}{2}$. Ok!

Inductive Step: If $Q(k) = \dfrac{k(k+1)}{2}$, then $Q(k+1) = \dfrac{(k+1)(k+2)}{2}$.

$Q(k+1) = Q(k) + (k+1)$ **Applying the recurrence**

$= \dfrac{k(k+1)}{2} + (k+1)$ **By the Inductive Hypothesis**

$= \dfrac{(k+1)(k+2)}{2}$ **After a bunch of algebra**

Therefore, $Q(n) = \dfrac{k(k+1)}{2}$

Extra Slides

Approximate Solutions to Rec. Relations

Theorem: (The Master Theorem) Given a recursive function of the form $T(n) = a \cdot T(n/b) + c \cdot n^2$, where:

$T(n)$ **is an increasing function,**

$n = b^k$**,**

k **is an integer** > 0**,**

a **is a real** ≥ 1

b **is an integer** > 1

c **is a real** > 0**, and**

d **is a real** ≥ 0**, then:**

$$f(n) = \begin{cases} O(n^d) & \textbf{if } a < b^d \\ O(n^d \cdot \log_2 n) & \textbf{if } a = < b^d \\ O(n^{\log_b a}) & \textbf{if } a > b^d \end{cases}$$

Proof: Rosen