Module #17:
Recurrence Relations

Rosen 5th ed., §6.1-6.3
~29 slides, ~1.5 lecture

§6.1: Recurrence Relations

- A *recurrence relation* (R.R., or just *recurrence*) for a sequence $\{a_n\}$ is an equation that expresses a_n in terms of one or more previous elements $a_0, \ldots, a_{n-1}$ of the sequence, for all $n \geq n_0$.
  - A recursive definition, without the base cases.
- A particular sequence (described non-recursively) is said to *solve* the given recurrence relation if it is consistent with the definition of the recurrence.
  - A given recurrence relation may have many solutions.

Recurrence Relation Example

- Consider the recurrence relation

$$a_n = 2a_{n-1} - a_{n-2} \quad (n \geq 2).$$

- Which of the following are solutions?

$a_n = 3n$ Yes

$a_n = 2^n$ No

$a_n = 5$ Yes

Example Applications

- Recurrence relation for growth of a bank account with *P*% interest per given period:

$$M_n = M_{n-1} + (P/100)M_{n-1}$$

- Growth of a population in which each organism yields 1 new one every period starting 2 periods after its birth.

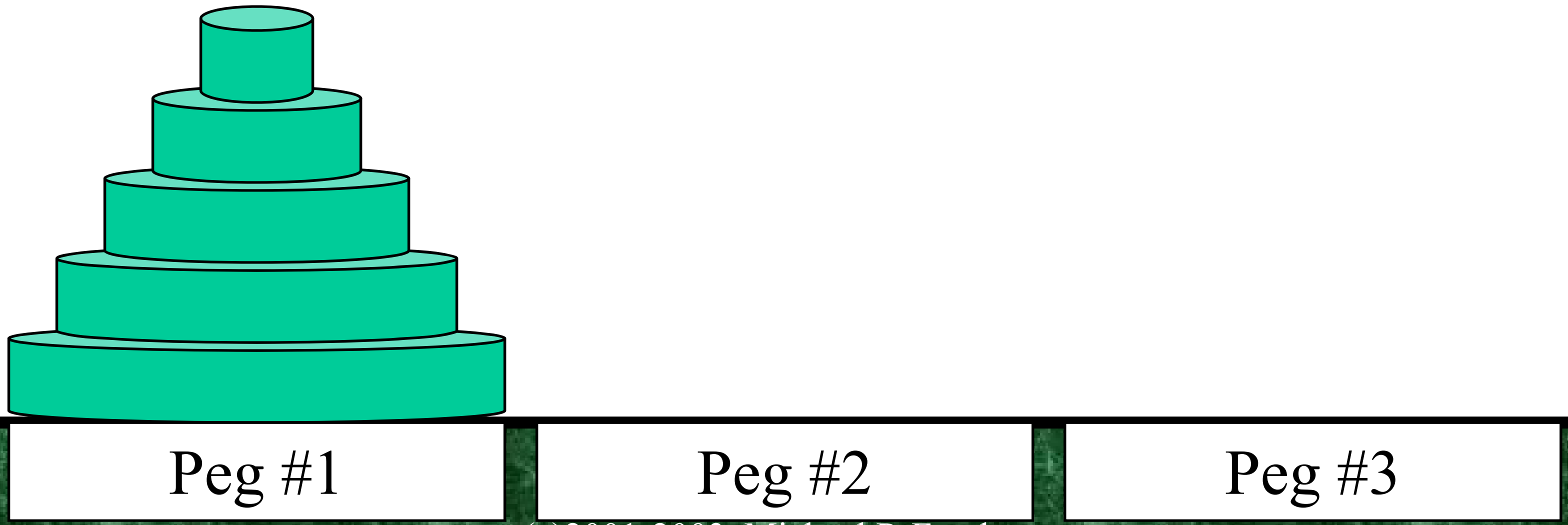
$$P_n = P_{n-1} + P_{n-2} \quad \text{(Fibonacci relation)}$$

Solving Compound Interest RR

- $M_n = M_{n-1} + (P/100)M_{n-1}$
  $= (1 + P/100)\ M_{n-1}$
  $= r\ M_{n-1}$ (let $r = 1 + P/100$)
  $= r\ (r\ M_{n-2})$
  $= r{\cdot}r{\cdot}(r\ M_{n-3})$ …and so on to…
  $= r^n\ M_0$

Tower of Hanoi Example

- Problem: Get all disks from peg 1 to peg 2.
  - Only move 1 disk at a time.
  - Never set a larger disk on a smaller one.



Hanoi Recurrence Relation

- Let H_n = # moves for a stack of n disks.
- Optimal strategy:
  - Move top $n-1$ disks to spare peg. (H_{n-1} moves)
  - Move bottom disk. (1 move)
  - Move top $n-1$ to bottom disk. (H_{n-1} moves)
- Note: $H_n = 2H_{n-1} + 1$

Solving Tower of Hanoi RR

$H_n = 2\,H_{n-1} + 1$

$= 2\,(2\,H_{n-2} + 1) + 1 \qquad = 2^2\,H_{n-2} + 2 + 1$

$= 2^2(2\,H_{n-3} + 1) + 2 + 1 \qquad = 2^3\,H_{n-3} + 2^2 + 2 + 1$

…

$= 2^{n-1}\,H_1 + 2^{n-2} + \ldots + 2 + 1$

$= 2^{n-1} + 2^{n-2} + \ldots + 2 + 1$ (since $H_1 = 1$)

$$= \sum_{i=0}^{n-1} 2^i$$

$= 2^n - 1$

§6.2: Solving Recurrences

General Solution Schemas

- A *linear homogeneous recurrence of degree k with constant coefficients* ("*k*-LiHoReCoCo") is a recurrence of the form

$$a_n = c_1 a_{n-1} + \ldots + c_k a_{n-k},$$

where the c_i are all real, and $c_k \neq 0$.
- The solution is uniquely determined if k initial conditions $a_0 \ldots a_{k-1}$ are provided.

Solving LiHoReCoCos

- Basic idea: Look for solutions of the form $a_n = r^n$, where r is a constant.
- This requires the *characteristic equation*:

$$r^n = c_1 r^{n-1} + \ldots + c_k r^{n-k},\ i.e.,$$
$$r^k - c_1 r^{k-1} - \ldots - c_k = 0$$

- The solutions (*characteristic roots*) can yield an explicit formula for the sequence.

Solving 2-LiHoReCoCos

- Consider an arbitrary 2-LiHoReCoCo:

  $$a_n = c_1 a_{n-1} + c_2 a_{n-2}$$

- It has the characteristic equation (C.E.):

  $$r^2 - c_1 r - c_2 = 0$$

- **Thm. 1:** If this CE has 2 roots $r_1 \neq r_2$, then

  $$a_n = \alpha_1 r_1^{\,n} + \alpha_2 r_2^{\,n} \text{ for } n \geq 0$$

  for some constants α_1, α_2.

Example

- Solve the recurrence $a_n = a_{n-1} + 2a_{n-2}$ given the initial conditions $a_0 = 2$, $a_1 = 7$.
- Solution: Use theorem 1
  - $c_1 = 1$, $c_2 = 2$
  - Characteristic equation: $r^2 - r - 2 = 0$
  - Solutions: $r = [-(-1)\pm((-1)^2 - 4\cdot 1\cdot(-2))^{1/2}] / 2\cdot 1$ $= (1\pm 9^{1/2})/2 = (1\pm 3)/2$, so $r = 2$ or $r = -1$.
  - So $a_n = \alpha_1\, 2^n + \alpha_2\, (-1)^n$.

Example Continued…

- To find α_1 and α_2, solve the equations for the initial conditions a_0 and a_1:

  $$a_0 = 2 = \alpha_1 2^0 + \alpha_2 (-1)^0$$
  $$a_1 = 7 = \alpha_1 2^1 + \alpha_2 (-1)^1$$

  Simplifying, we have the pair of equations:

  $$2 = \alpha_1 + \alpha_2$$
  $$7 = 2\alpha_1 - \alpha_2$$

  which we can solve easily by substitution:

  $$\alpha_2 = 2-\alpha_1; \quad 7 = 2\alpha_1 - (2-\alpha_1) = 3\alpha_1 - 2;$$
  $$9 = 3\alpha_1; \quad \alpha_1 = 3; \quad \alpha_2 = 1.$$

- Final answer: $a_n = 3{\cdot}2^n - (-1)^n$

Check: $\{a_{n\geq 0}\} = 2, 7, 11, 25, 47, 97 \ldots$

The Case of Degenerate Roots

- Now, what if the C.E. $r^2 - c_1r - c_2 = 0$ has only 1 root r_0?
- **Theorem 2:** Then,

$$a_n = \alpha_1 r_0{}^n + \alpha_2 n r_0{}^n, \quad \text{for all } n \geq 0,$$

for some constants α_1, α_2.

k-LiHoReCoCos

- Consider a *k*-LiHoReCoCo: $$a_n = \sum_{i=1}^{k} c_i a_{n-i}$$
- It's C.E. is: $$r^k - \sum_{i=1}^{k} c_i r^{k-i} = 0$$
- **Thm.3:** If this has *k* distinct roots r_i, then the solutions to the recurrence are of the form:

$$a_n = \sum_{i=1}^{k} \alpha_i r_i^n$$

for all $n \geq 0$, where the α_i are constants.

Degenerate *k*-LiHoReCoCos

- Suppose there are t roots $r_1,\ldots,r_t$ with multiplicities $m_1,\ldots,m_t$. Then:

$$a_n = \sum_{i=1}^{t}\left(\sum_{j=0}^{m_i-1}\alpha_{i,j}n^j\right)r_i^n$$

for all $n\geq 0$, where all the α are constants.

Li**<u>No</u>**ReCoCos

- Linear *<u>no</u>nhomogeneous* RRs with constant coefficients may (unlike Li**<u>Ho</u>**ReCoCos) contain some terms $F(n)$ that depend *only* on n (and *not* on any a_i's). General form:

$$\underbrace{a_n = c_1 a_{n-1} + \ldots + c_k a_{n-k}} + F(n)$$

The *associated homogeneous recurrence relation* (associated Li<u>Ho</u>ReCoCo).

Solutions of LiNoReCoCos

- A useful theorem about LiNoReCoCos:
  - If $a_n = p(n)$ is any *particular* solution to the LiNoReCoCo
  $$a_n = \left(\sum_{i=1}^{k} c_i a_{n-i}\right) + F(n)$$
  - Then *all* its solutions are of the form:
  $$a_n = p(n) + h(n),$$
  where $a_n = h(n)$ is any solution to the associated homogeneous RR
  $$a_n = \left(\sum_{i=1}^{k} c_i a_{n-i}\right)$$

Example

- Find all solutions to $a_n = 3a_{n-1}+2n$. Which solution has $a_1 = 3$?
  - Notice this is a 1-LiNoReCoCo. Its associated 1-LiHoReCoCo is $a_n = 3a_{n-1}$, whose solutions are all of the form $a_n = \alpha 3^n$. Thus the solutions to the original problem are all of the form $a_n = p(n) + \alpha 3^n$. So, all we need to do is find one $p(n)$ that works.

Trial Solutions

- If the extra terms $F(n)$ are a degree-t polynomial in n, you should try a degree-t polynomial as the particular solution $p(n)$.
- This case: $F(n)$ is linear so try $a_n = cn + d$.

  $cn+d = 3(c(n-1)+d) + 2n$ (for all n)
  $(-2c+2)n + (3c-2d) = 0$ (collect terms)
  So $c = -1$ and $d = -3/2$.

  So $a_n = -n - 3/2$ is a solution.
- Check: $a_{n\geq 1} = \{-5/2, -7/2, -9/2, \ldots\}$

Finding a Desired Solution

- From the previous, we know that all general solutions to our example are of the form:

  $$a_n = -n - 3/2 + \alpha 3^n.$$

  Solve this for α for the given case, $a_1 = 3$:

  $$3 = -1 - 3/2 + \alpha 3^1$$

  $$\alpha = 11/6$$

- The answer is $a_n = -n - 3/2 + (11/6)3^n$

§5.3: Divide & Conquer R.R.s

Main points so far:

- Many types of problems are solvable by reducing a problem of size n into some number a of independent subproblems, each of size $\leq\lceil n/b\rceil$, where $a\geq 1$ and $b>1$.
- The time complexity to solve such problems is given by a recurrence relation:
  - $T(n) = a\,T(\lceil n/b\rceil) + g(n)$

Time for each subproblem

Time to break problem up into subproblems

Divide+Conquer Examples

- **Binary search:** Break list into 1 sub-problem (smaller list) (so a=1) of size $\leq\lceil n/2 \rceil$ (so b=2).
  - So $T(n) = T(\lceil n/2 \rceil)+c$ ($g(n)=c$ constant)
- **Merge sort:** Break list of length n into 2 sublists (a=2), each of size $\leq\lceil n/2 \rceil$ (so b=2), then merge them, in $g(n) = \Theta(n)$ time.
  - So $T(n) = T(\lceil n/2 \rceil) + cn$ (roughly, for some c)

Fast Multiplication Example

- The ordinary grade-school algorithm takes $\Theta(n^2)$ steps to multiply two n-digit numbers.
  - This seems like too much work!
- So, let's find an asymptotically *faster* multiplication algorithm!
- To find the product cd of two $2n$-digit base-b numbers, $c=(c_{2n-1}c_{2n-2}\ldots c_0)_b$ and $d=(d_{2n-1}d_{2n-2}\ldots d_0)_b$, first, we break c and d in half:

  $c=b^nC_1+C_0, \qquad d=b^nD_1+D_0,$

  and then... (see next slide)

Derivation of Fast Multiplication

$$cd = (b^n C_1 + C_0)(b^n D_1 + D_0)$$

$$= b^{2n} C_1 D_1 + b^n (C_1 D_0 + C_0 D_1) + C_0 D_0$$

(Multiply out polynomials)

$$= b^{2n} C_1 D_1 + C_0 D_0 +$$

$$b^n (C_1 D_0 + C_0 D_1 + (C_1 D_1 - C_1 D_1) + (C_0 D_0 - C_0 D_0))$$

Zero

$$= (b^{2n} + b^n) C_1 D_1 + (b^n + 1) C_0 D_0 +$$

$$b^n (C_1 D_0 - C_1 D_1 - C_0 D_0 + C_0 D_1)$$

$$= (b^{2n} + b^n) C_1 D_1 + (b^n + 1) C_0 D_0 +$$

$$b^n (C_1 - C_0)(D_0 - D_1)$$

(Factor last polynomial)

Three multiplications, each with n-digit numbers

Recurrence Rel. for Fast Mult.

Notice that the time complexity $T(n)$ of the fast multiplication algorithm obeys the recurrence:

- $T(2n)=3T(n)+\Theta(n)$

  i.e.,

- $T(n)=3T(n/2)+\Theta(n)$

  So $a=3$, $b=2$.

Time to do the needed adds & subtracts of n-digit and $2n$-digit numbers

The Master Theorem

Consider a function $f(n)$ that, for all $n=b^k$ for all $k\in\mathbf{Z}^+$, satisfies the recurrence relation:

$$f(n) = af(n/b) + cn^d$$

with $a\geq 1$, integer $b>1$, real $c>0$, $d\geq 0$. Then:

$$f(n) \in \begin{cases} O(n^d) & \text{if } a < b^d \\ O(n^d \log n) & \text{if } a = b^d \\ O(n^{\log_b a}) & \text{if } a > b^d \end{cases}$$

Master Theorem Example

- Recall that complexity of fast multiply was:

  $T(n)=3T(n/2)+\Theta(n)$

- Thus, $a=3$, $b=2$, $d=1$. So $a > b^d$, so case 3 of the master theorem applies, so:

  $$T(n) = O(n^{\log_b a}) = O(n^{\log_2 3})$$

  which is $O(n^{1.58\ldots})$, so the new algorithm is strictly faster than ordinary $\Theta(n^2)$ multiply!

§6.4: Generating Functions

- Not covered this semester.

§6.5: Inclusion-Exclusion

- This topic will have been covered out-of-order already in Module #15, Combinatorics.
- As for Section 6.6, applications of Inclusion-Exclusion: No slides yet.