Advanced Combinatorics: Recurrence Relations

Section 8.1

Section Summary

- Applications of Recurrence Relations
  - Fibonacci Numbers
  - The Tower of Hanoi
  - Counting Problems
- Algorithms and Recurrence Relations (*not currently included in overheads*)

Recurrence Relations
(recalling definitions from Chapter 2)

Definition: A *recurrence relation* for the sequence $\{a_n\}$ is an equation that expresses a_n in terms of one or more of the previous terms of the sequence, namely, $a_0, a_1, \ldots, a_{n-1}$, for all integers n with $n \geq n_0$, where n_0 is a nonnegative integer.

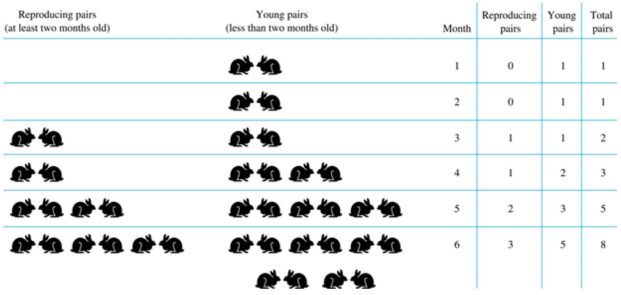
- A sequence is called a *solution* of a recurrence relation if its terms satisfy the recurrence relation.
- The *initial conditions* for a sequence specify the terms that precede the first term where the recurrence relation takes effect.

Rabbits and the Fibonacci Numbers

Example: A young pair of rabbits (one of each gender) is placed on an island. A pair of rabbits does not breed until they are 2 months old. After they are 2 months old, each pair of rabbits produces another pair each month. Find a recurrence relation for the number of pairs of rabbits on the island after *n* months, assuming that rabbits never die.

This is the original problem considered by Leonardo Pisano (Fibonacci) in the thirteenth century.

Rabbits and the Fibonacci Numbers (*cont.*)



Reproducing pairs (at least two months old)	Young pairs (less than two months old)	Month	Reproducing pairs	Young pairs	Total pairs
		1	0	1	1
		2	0	1	1
		3	1	1	2
		4	1	2	3
		5	2	3	5
		6	3	5	8

Modeling the Population Growth of Rabbits on an Island

Rabbits and the Fibonacci Numbers (*cont.*)

Solution: Let f_n be the the number of pairs of rabbits after n months.

- There are is $f_1 = 1$ pairs of rabbits on the island at the end of the first month.
- We also have $f_2 = 1$ because the pair does not breed during the first month.
- To find the number of pairs on the island after n months, add the number on the island after the previous month, f_{n-1}, and the number of newborn pairs, which equals f_{n-2}, because each newborn pair comes from a pair at least two months old.

Consequently the sequence $\{f_n\}$ satisfies the recurrence relation $f_n = f_{n-1} + f_{n-2}$ for $n \geq 3$ with the initial conditions $f_1 = 1$ and $f_2 = 1$.

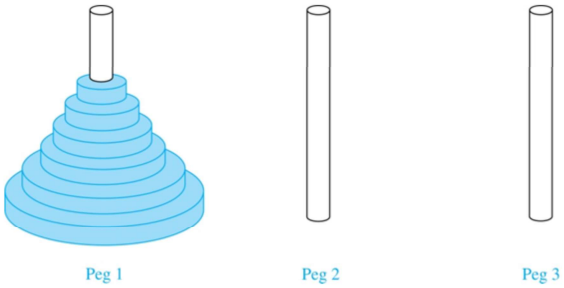
The number of pairs of rabbits on the island after n months is given by the nth Fibonacci number.

The Tower of Hanoi

In the late nineteenth century, the French mathematician Édouard Lucas invented a puzzle consisting of three pegs on a board with disks of different sizes. Initially all of the disks are on the first peg in order of size, with the largest on the bottom.

- **Rules:** You are allowed to move the disks one at a time from one peg to another as long as a larger disk is never placed on a smaller.
- **Goal:** Using allowable moves, end up with all the disks on the second peg in order of size with largest on the bottom.

The Tower of Hanoi (*continued*)



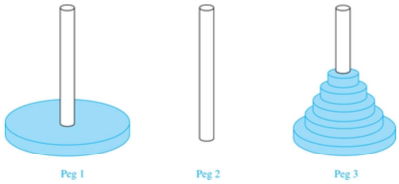
The Initial Position in the Tower of Hanoi Puzzle
https://youtu.be/2SUvWfNJSsM

The Tower of Hanoi (*continued*)

Solution: Let $\{H_n\}$ denote the number of moves to solve the Tower of Hanoi with n disks. Set up a recurrence relation for the sequence $\{H_n\}$. Begin with n disks on peg 1. We can transfer the top $n-1$ disks, following the rules of the puzzle, to peg 3 using H_{n-1} moves.

First, use 1 move to transfer the largest disk to the second peg. Then transfer the $n-1$ disks from peg 3 to peg 2 using H_{n-1} additional moves. (Cannot be done faster). Hence, $H_n = 2H_{n-1} + 1$.

The initial condition is $H_1 = 1$ since a single disk can be transferred from peg 1 to peg 2 in one move.



The Tower of Hanoi (*continued*)

Use iterations to solve this recurrence relation by repeatedly expressing H_n in terms of the previous terms of the sequence.

$H_n = 2H_{n-1} + 1 = 2(2H_{n-2} + 1) + 1 = 2^2 H_{n-2} + 2 + 1$

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

$\vdots$

$= 2^{n-1}H_1 + 2^{n-2} + 2^{n-3} + \ldots + 2 + 1$

$= 2^{n-1} + 2^{n-2} + 2^{n-3} + \ldots + 2 + 1$ *because* $H_1 = 1$

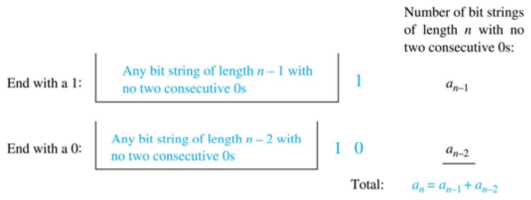
$= 2^n - 1$ *using the formula for the sum of the terms of a geometric series*

- A Myth: Monks in a tower in Hanoi are transferring 64 gold disks from one peg to another following the rules of the puzzle. They move one disk each day. When the puzzle is finished, the world will end.
- Using this formula for the 64 gold disks of the myth, $2^{64} - 1 = 18{,}446{,}744{,}073{,}709{,}551{,}615$ days are needed to solve the puzzle, which is more than 500 billion years.
- Reve's puzzle (proposed in 1907 by Henry Dudeney) has 4 pegs. There is a well-known unsolved conjecture for the the minimum number of moves needed to solve this puzzle. (*see Exercises* 38-45)

Counting Bit Strings

Example 3: Find a recurrence relation and give initial conditions for the number of bit strings of length n without two consecutive 0s. How many such bit strings are there of length five?

Solution: Let a_n denote the number of bit strings of length n without two consecutive 0s. To obtain a recurrence relation for $\{a_n\}$ note that the number of bit strings of length n that do not have two consecutive 0s is the number of bit strings ending with a 0 plus the number of such bit strings ending with a 1.



Now assume that $n \geq 3$.

- The bit strings of length n ending with 1 without two consecutive 0s are the bit strings of length $n-1$ with no two consecutive 0s with a 1 at the end. Hence, there are a_{n-1} such bit strings.
- The bit strings of length n ending with 0 without two consecutive 0s are the bit strings of length $n-2$ with no two consecutive 0s with 10 at the end. Hence, there are a_{n-2} such bit strings.

Conclude that $a_n = a_{n-1} + a_{n-2}$ for $n \geq 3$.

Bit Strings (*continued*)

The initial conditions are:

- $a_1 = 2$, since both the bit strings 0 and 1 do not have consecutive 0s.
- $a_2 = 3$, since the bit strings 01, 10, and 11 do not have consecutive 0s, while 00 does.

To obtain a_5, use the recurrence relation three times:

- $a_3 = a_2 + a_1 = 3 + 2 = 5$
- $a_4 = a_3 + a_2 = 5 + 3 = 8$
- $a_5 = a_4 + a_3 = 8 + 5 = 13$

- Note that $\{a_n\}$ satisfies the same recurrence relation as the Fibonacci sequence. Since $a_1 = f_3$ and $a_2 = f_4$, we conclude that $a_n = f_{n+2}$.

Counting the Ways to Parenthesize a Product

Example: Find a recurrence relation for C_n, to parenthesize the product of $n+1$ numbers, $x_0 \cdot x_1 \cdot x_2 \cdot \cdots \cdot x_n$, to specify the order of multiplication. For example, $C_3 = 5$, since all the possible ways to parenthesize 4 numbers are

$((x_0 \cdot x_1) \cdot x_2) \cdot x_3$, $(x_0 \cdot (x_1 \cdot x_2)) \cdot x_3$, $(x_0 \cdot x_1) \cdot (x_2 \cdot x_3)$, $x_0 \cdot ((x_1 \cdot x_2) \cdot x_3)$, $x_0 \cdot (x_1 \cdot (x_2 \cdot x_3))$

Solution: Note that however parentheses are inserted in $x_0 \cdot x_1 \cdot x_2 \cdot \cdots \cdot x_n$, one "$\cdot$" operator remains outside all parentheses. This final operator appears between two of the $n+1$ numbers, say x_k and x_{k+1}. Since there are C_k ways to insert parentheses in the product $x_0 \cdot x_1 \cdot x_2 \cdot \cdots \cdot x_k$ and C_{n-k-1} ways to insert parentheses in the product $x_{k+1} \cdot x_{k+2} \cdot \cdots \cdot x_n$, we have

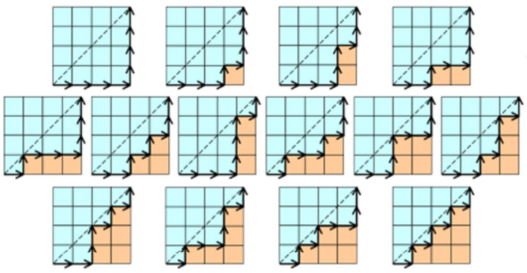
$$\begin{aligned} C_n &= C_0 C_{n-1} + C_1 C_{n-2} + \cdots + C_{n-2} C_1 + C_{n-1} C_0 \\ &= \sum_{k=0}^{n-1} C_k C_{n-k-1} \end{aligned}$$

The initial conditions are $C_0 = 1$ and $C_1 = 1$.

Catalan Numbers

- The sequence $\{C_n\}$ is **Catalan Numbers**. $C_n = \frac{1}{n+1}\binom{2n}{n}$

n	0	1	2	3	4	5	6	7	8	9	10
C_n	1	1	2	5	14	42	132	429	1430	4862	16796



((((()))) ((()())) ((())()) (()(()))

()((())) (()()()) ((()))() (())(()) (()())()

(())()() ()(())() ()()(()) ()()()()

ABCDE**** ABCD*E*** ABCD**E** ABC*DE***

AB*CDE*** ABC*D*E** ABCD***E* ABC**DE** ABC*D**E*

ABC**D*E* AB*CD**E* AB*C*DE** AB*C*D*E*

1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796,

Reverse Polish Notation / Postfix Notation

- Jan Łukasiewicz (1924).
  Infix: (2+3)*11+1
  Postfix: 2 3 add 11 mul 1 add
  Prefix/Coq: (add (mul (add 2 3) 11) 1)

Input	2	3	add	11	mul	1	add
Stack	2	3 2	5	11 5	55	1 55	56

ABCDE**** ABCD*E*** ABCD**E** ABC*DE***

AB*CDE*** ABC*D*E** ABCD***E* ABC**DE** ABC*D**E*

ABC**D*E* AB*CD**E* AB*C*DE** AB*C*D*E*

ABCD***E*
(A*(B*(C*D)))*E