UNIVERSITY AT ALBANY
State University of New York

Lecture 9: Advanced Counting Techniques

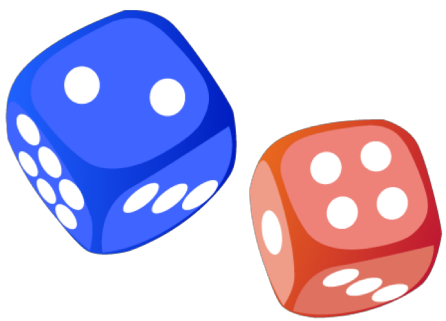
Dr. Chengjiang Long
Computer Vision Researcher at Kitware Inc.
Adjunct Professor at SUNY at Albany.
Email: **clong2@albany.edu**

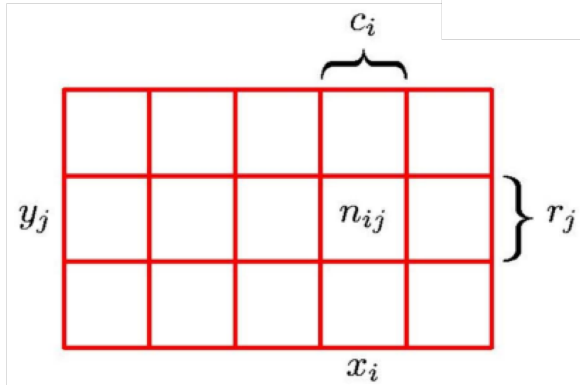
Recap Previous Lecture

- Discrete Probability
- Conditional Probability

Joint Probability

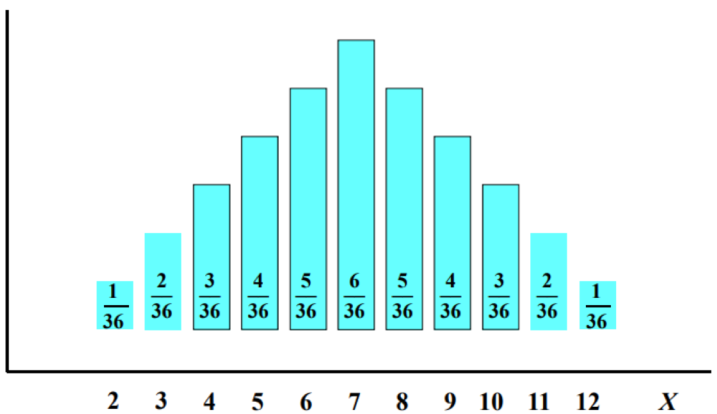
$$p(X = x_i, Y = y_j) = \frac{n_{ij}}{N}$$





Conditional Probability

$$p(Y = y_j | X = x_i) = \frac{n_{ij}}{c_i}$$



Marginal Probability

$$p(X = x_i) = \frac{c_i}{N}.$$

Recap Previous Lecture

- Bayes Rules, Expected Value, Variances
- Binominal Distribution

$$\binom{n}{X} p^X (1-p)^{n-X}$$

$$P(x \mid y) = \frac{P(x,y)}{P(y)} = \frac{P(y \mid x)P(x)}{\sum_{x \in X} P(x,y)}$$

$$\text{posterior} = \frac{\text{likelihood} * \text{prior}}{\text{evidence}}$$

$$E(X) = \sum_{\text{all } x} x_i p(x_i)$$

$$E(X) = \int_{\text{all } x} x_i p(x_i) dx$$

$$\sigma^2 = Var(x) = E[(x-\mu)^2] = \sum_{\text{all } x} (x_i - \mu)^2 p(x_i)$$

Outline

- Applications of Recurrence Relations
- Solving Linear Recurrence Relations
- Divide-and-Conquer Algorithms and Recurrence Relations
- Generating Functions

Applications of Recurrence Relations

Recurrence Relations

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)

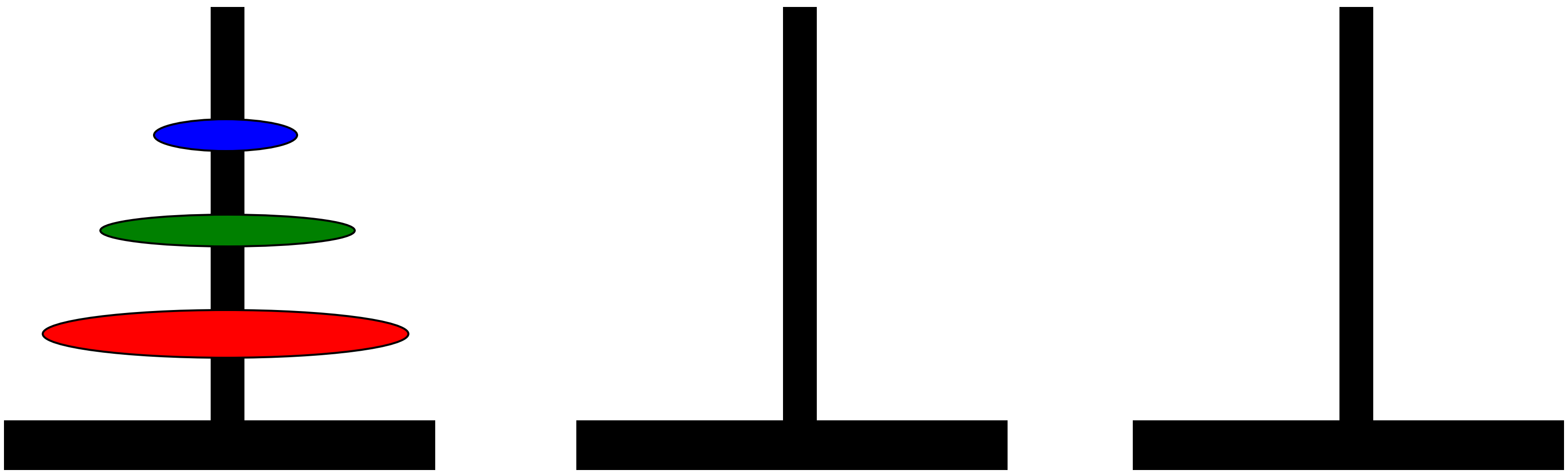
Solution: Let f_n be the the number of pairs of rabbits after n months.

- 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.

Towers of Hanoi (N=3)



- There are three pegs.
- 3 gold disks, with decreasing sizes, placed on the first peg.
- You need to move all of the disks from the first peg to the second peg.
- Larger disks cannot be placed on top of smaller disks.
- The third peg can be used to temporarily hold disks.

Towers of Hanoi

- The disks must be moved within one week. Assume one disk can be moved in 1 second. Is this possible?
- To create an algorithm to solve this problem, it is convenient to generalize the problem to the "N-disk" problem, where in our case N = 64.

Towers of Hanoi

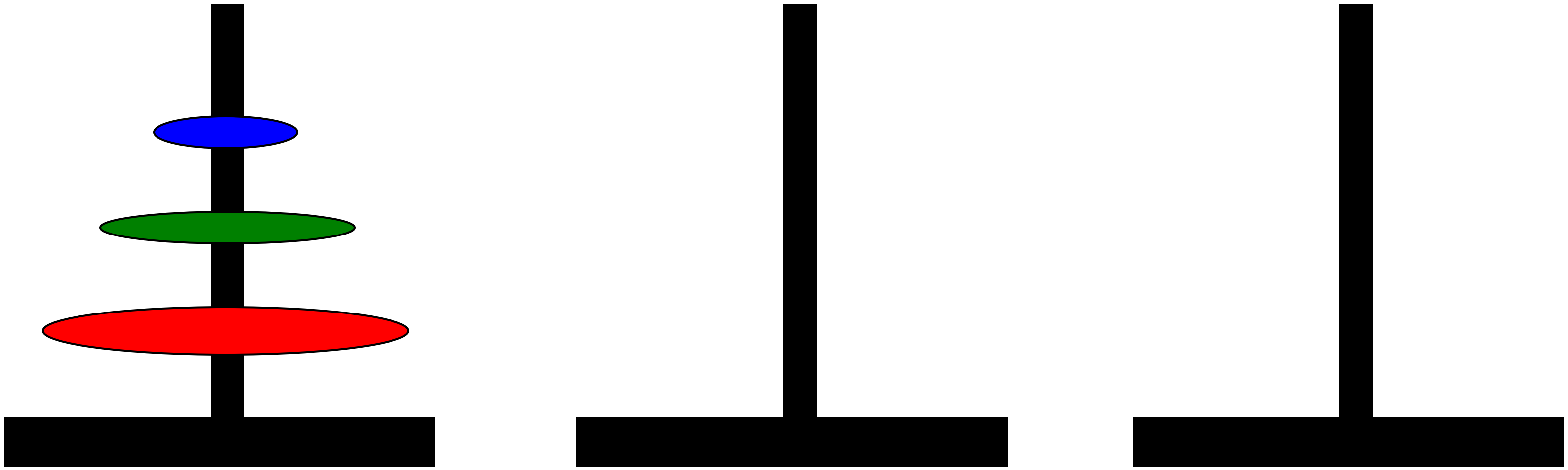
- How to solve it?
- Think recursively!!!!
- Suppose you could solve the problem for n-1 disks, i.e., you can move (n-1) disks from one tower to another, without ever having a large disk on top of a smaller disk. How would you do it for n?

Towers of Hanoi

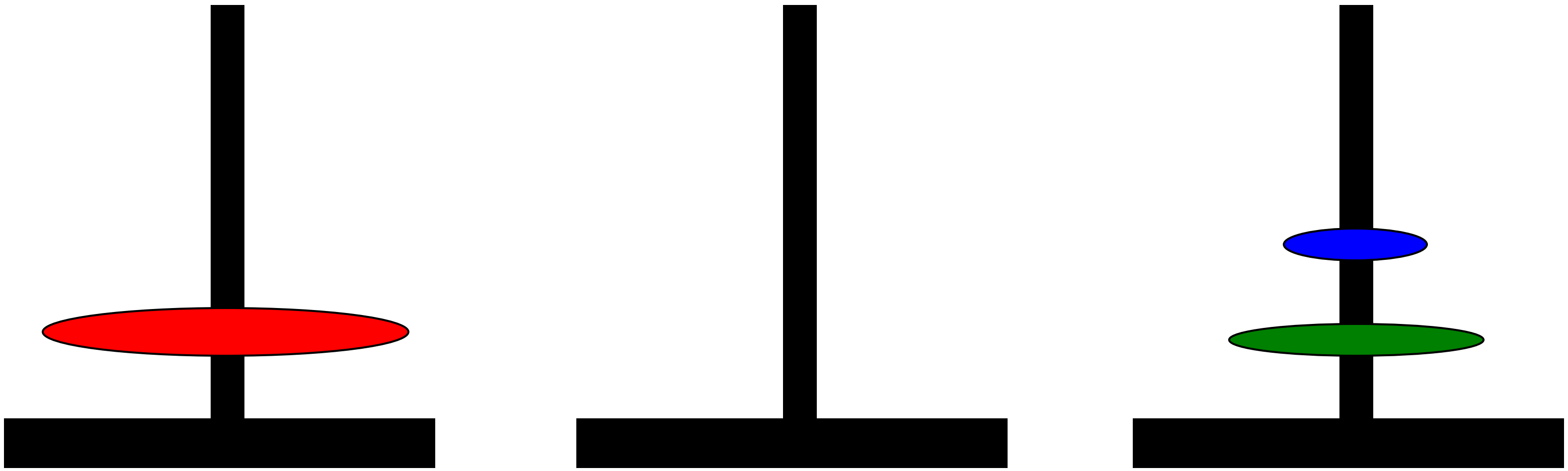
- Solution:

1. Move top (n-1) disks from tower 1 to tower 3 (you can do this by assumption – just pretend the largest ring is not there at all).
2. Move largest ring from tower 1 to tower 2.
3. Move top (n-1) rings from tower 3 to tower 2 (again, you can do this by assumption).

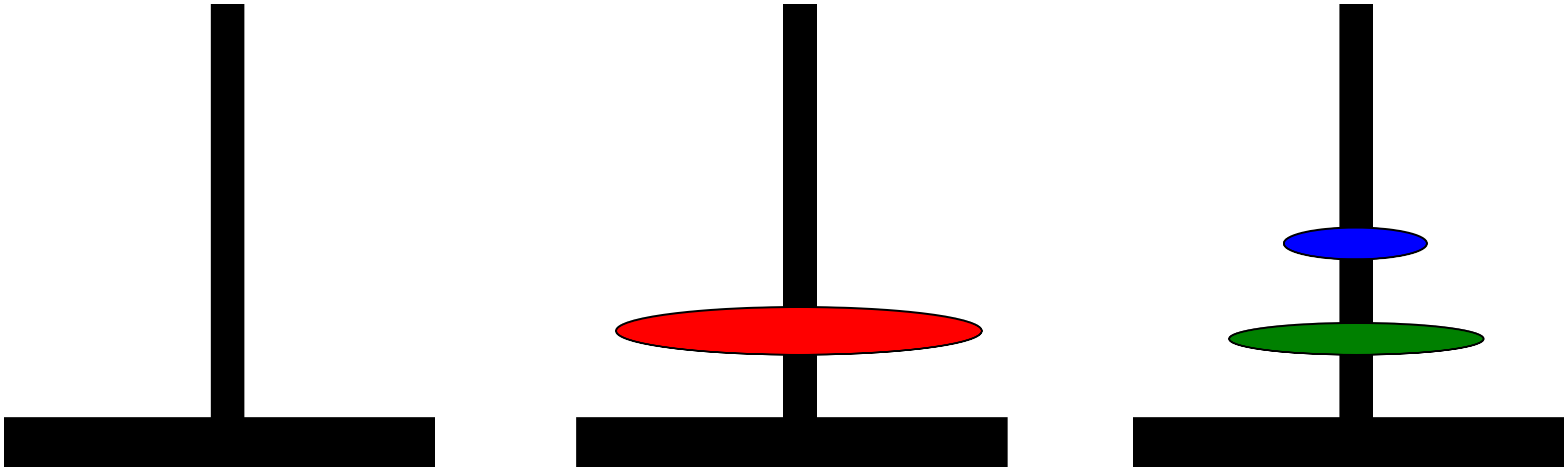
Recursive Solution



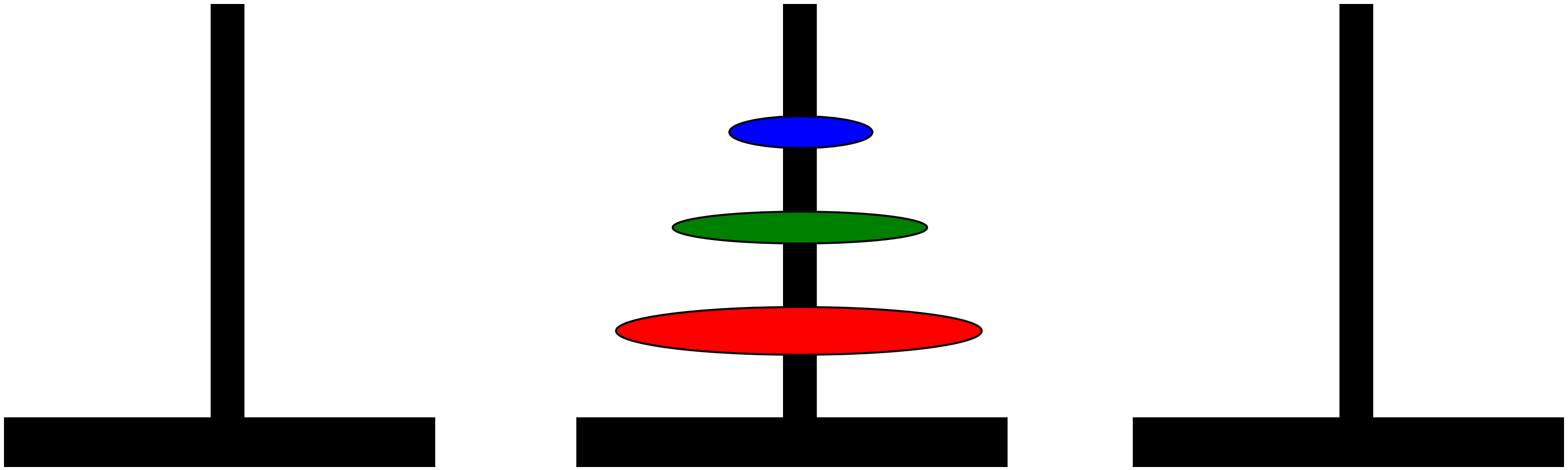
Recursive Solution



Recursive Solution



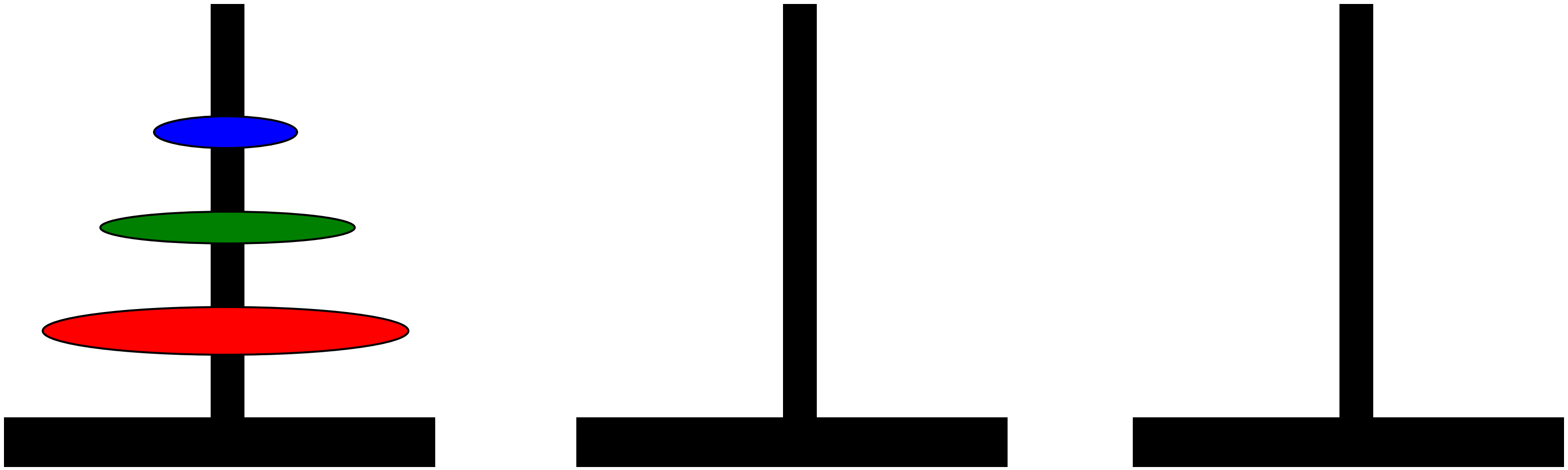
Recursive Solution



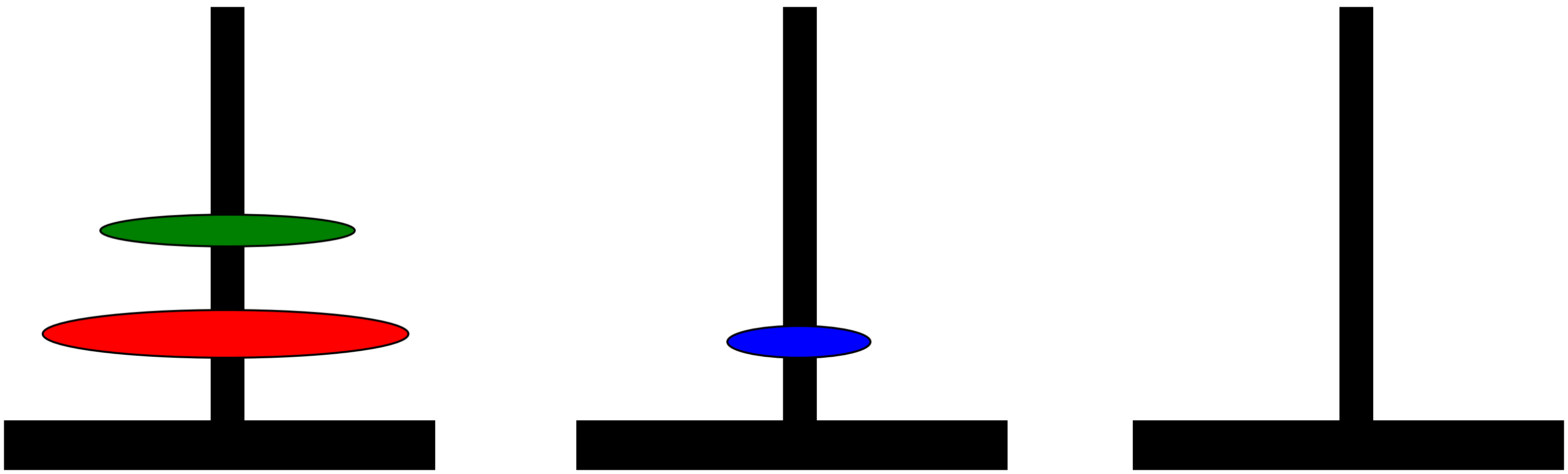
Towers of Hanoi

- **Procedure** *TowerHanoi* (n, a, b, c: integers, $1 \le a \le 3$, $1 \le b \le 3$, $1 \le c \le 3$)
- **if** n= 1 **then**
- *move*(a,b)
- **else**
- **begin**
- *TowerHanoi(*n-1, a, c, b)
- *move*(a,b);
- *TowerHanoi* (n-1,c,b,a);
- **end**

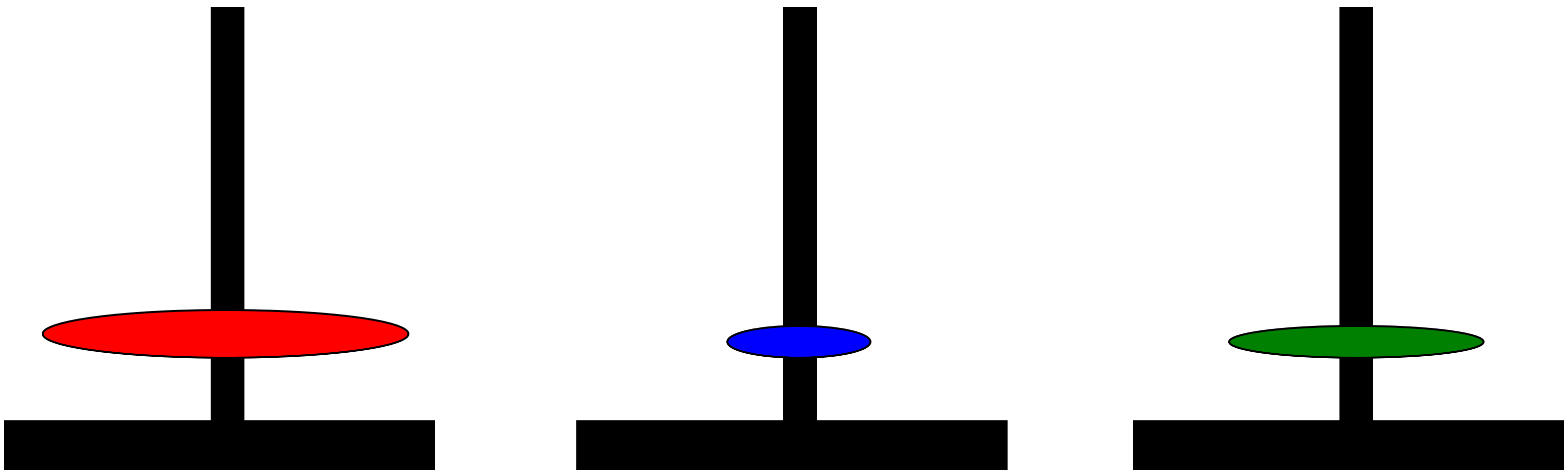
Towers of Hanoi



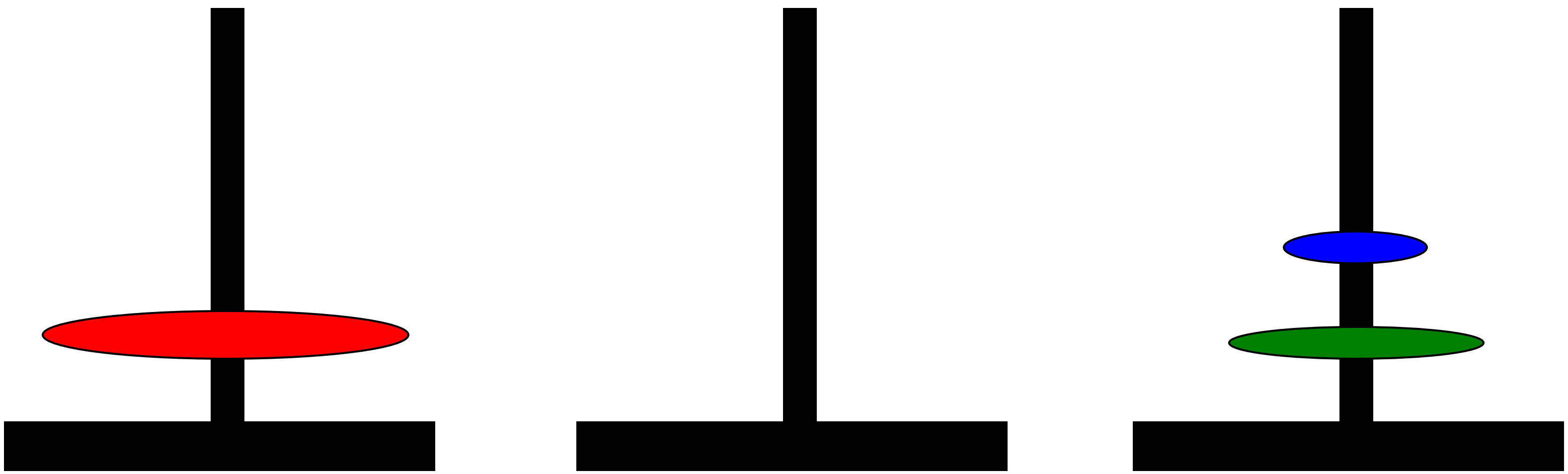
Towers of Hanoi



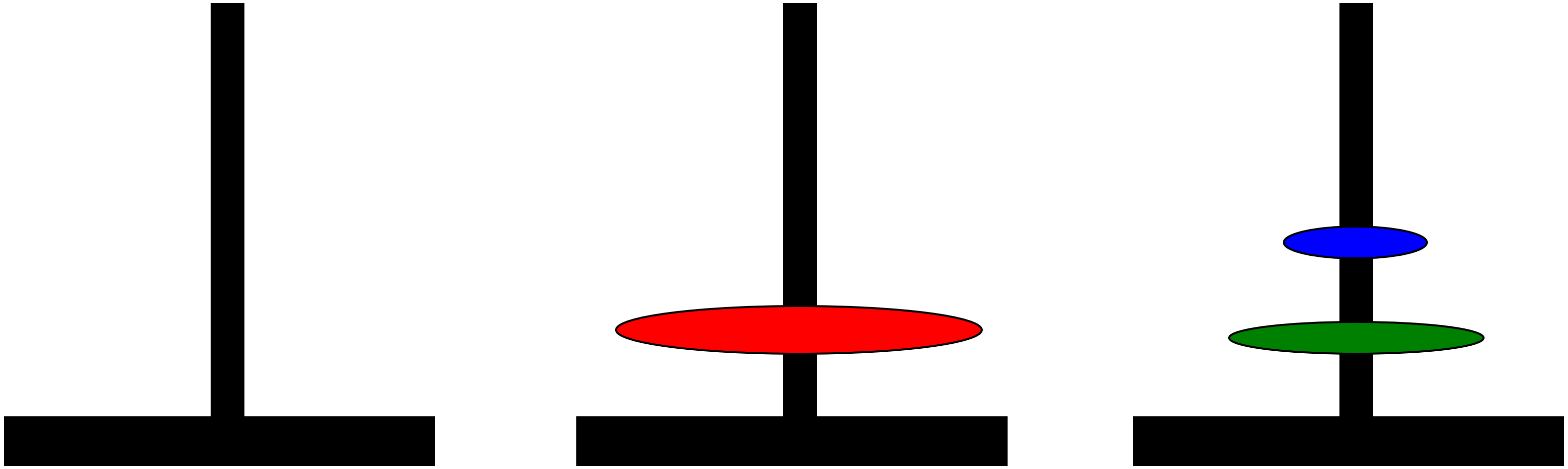
Towers of Hanoi



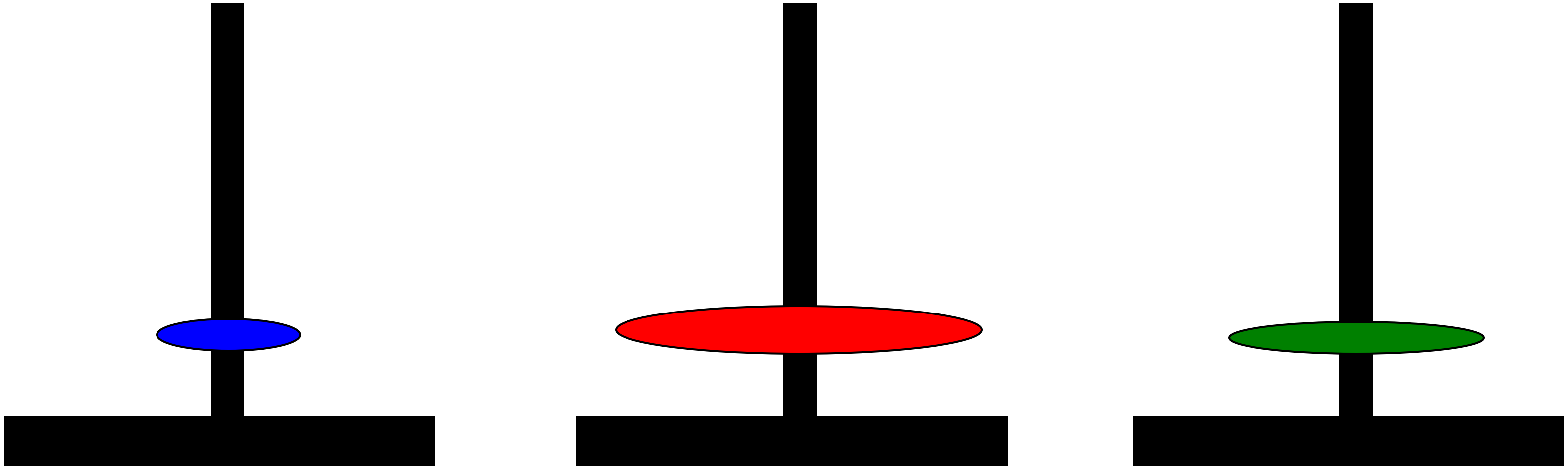
Towers of Hanoi



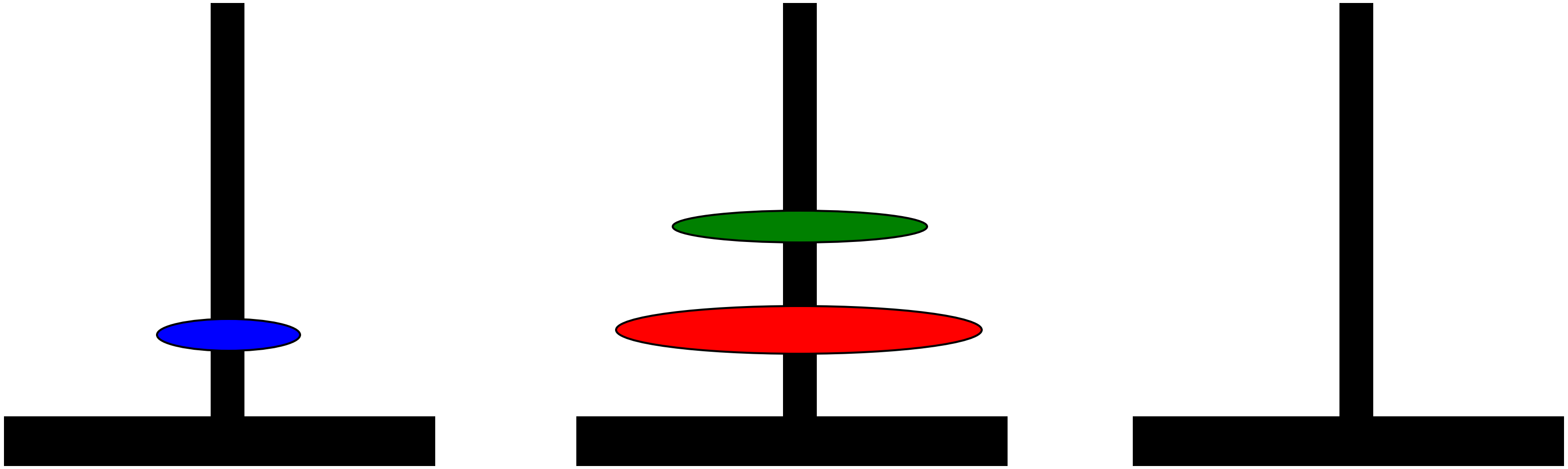
Towers of Hanoi



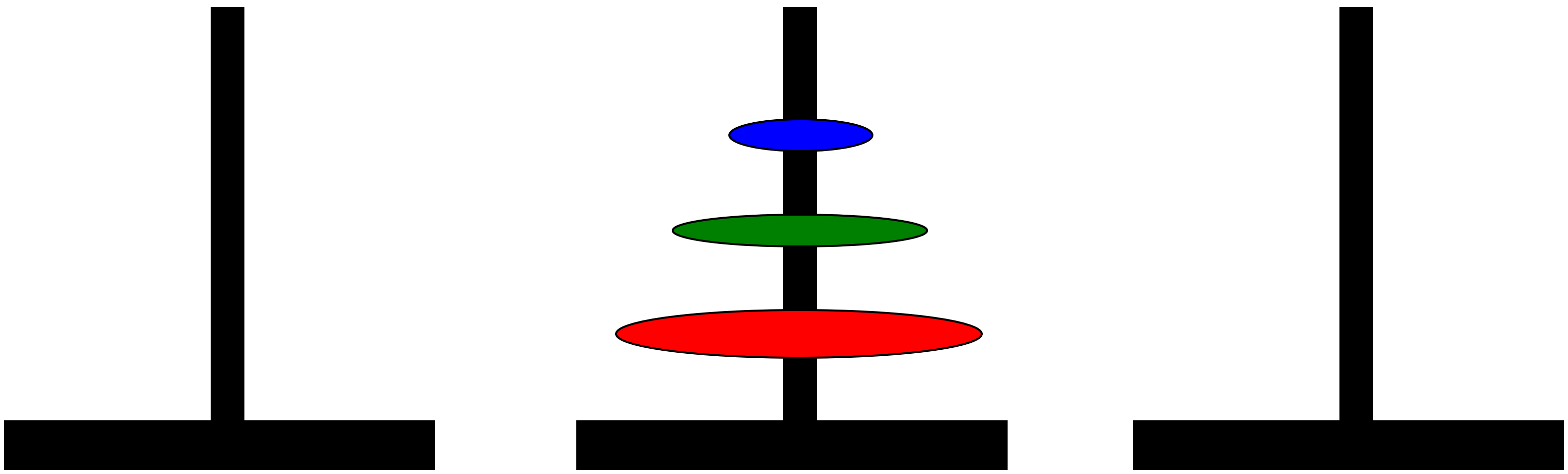
Towers of Hanoi



Towers of Hanoi



Towers of Hanoi



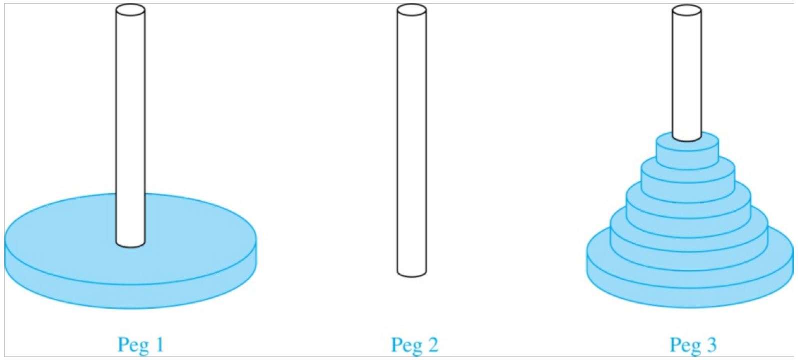
The Tower of Hanoi (*continued*)

- We can use an iterative approach to solve this recurrence relation by repeatedly expressing H_n in terms of the previous terms of the sequence.

$$
\begin{aligned}
H_n &= 2H_{n-1} + 1 \\
&= 2(2H_{n-2} + 1) + 1 = 2^2 H_{n-2} + 2 + 1 \\
&= 2^2(2H_{n-3} + 1) + 2 + 1 = 2^3 H_{n-3} + 2^2 + 2 + 1 \\
&\vdots \\
&= 2^{n-1}H_1 + 2^{n-2} + 2^{n-3} + \ldots + 2 + 1 \\
&= 2^{n-1} + 2^{n-2} + 2^{n-3} + \ldots + 2 + 1 \quad \textit{because } H_1 = 1 \\
&= 2^n - 1 \quad \textit{using the formula for the sum of the terms of a geometric series}
\end{aligned}
$$

The Tower of Hanoi (*continued*)

- **Solution**: Let $\{H_n\}$ denote the number of moves needed to solve the Tower of Hanoi Puzzle 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, we use 1 move to transfer the largest disk to the second peg. Then we transfer the $n - 1$ disks from peg 3 to peg 2 using H_{n-1} additional moves. This can not be done in fewer steps. 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.

Counting Bit Strings

Example: 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?

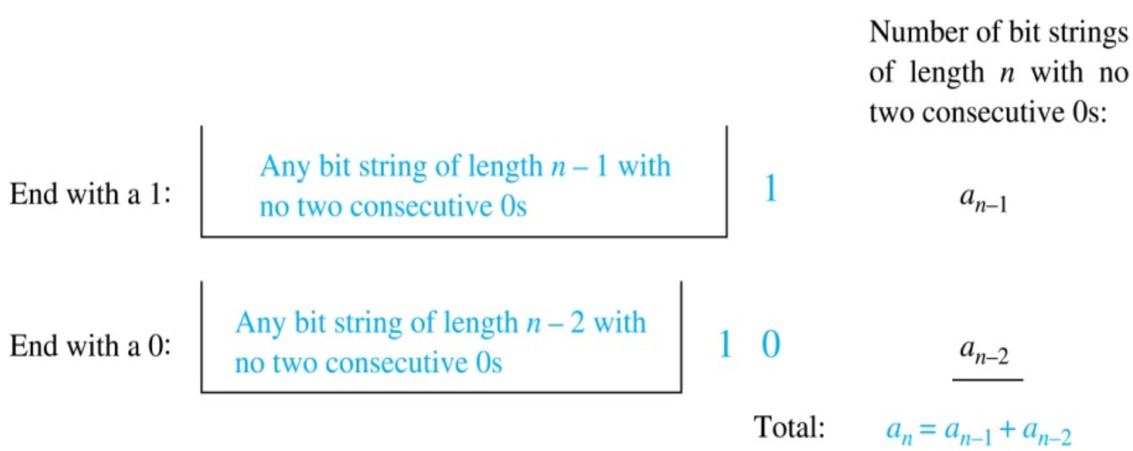
Solution: Let a_n denote the number of bit strings of length n without two consecutive 0s. To obtain a recurrence relation for $\{a_n\}$ note that the number of bit strings of length n that do not have two consecutive 0s is the number of bit strings ending with a 0 plus the number of such bit strings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

Counting Bit Strings (cont)

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.

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



Bit Strings (*cont*)

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, we use the recurrence relation three times to find that:

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

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 $a_n = f_{n+2}$.

Solving Linear Recurrence Relations

Linear Homogeneous Recurrence Relations

Definition: A *linear homogeneous recurrence relation of degree* k *with constant coefficients* is a recurrence relation of the form $a_n = c_1 a_{n-1} + c_2 a_{n-2} + \ldots\ldots + c_k a_{n-k}$, where $c_1, c_2, \ldots\ldots, c_k$ are real numbers, and $c_k \neq 0$

- it is *linear* because the right-hand side is a sum of the previous terms of the sequence each multiplied by a function of *n*.
- it is *homogeneous* because no terms occur that are not multiples of the a_js. Each coefficient is a constant.
- the *degree* is *k* because a_n is expressed in terms of the previous *k* terms of the sequence.

Linear Homogeneous Recurrence Relations

Definition: A *linear homogeneous recurrence relation of degree* k *with constant coefficients* is a recurrence relation of the form $a_n = c_1a_{n-1} + c_2a_{n-2} + \ldots.. + c_k a_{n-k}$, where $c_1, c_2, \ldots..,c_k$ are real numbers, and $c_k \neq 0$

By strong induction, a sequence satisfying such a recurrence relation is uniquely determined by the recurrence relation and the *k* initial conditions $a_0 = C_1, a_0 = C_1 ,\ldots , a_{k-1} = C_{k-1}$.

Examples of Linear Homogeneous Recurrence Relations

- $P_n = (1.11)P_{n-1}$ linear homogeneous recurrence relation of degree one
- $f_n = f_{n-1} + f_{n-2}$ linear homogeneous recurrence relation of degree two
- $a_n = a_{n-1} + a_{n-2}^2$ not linear
- $H_n = 2H_{n-1} + 1$ not homogeneous
- $B_n = nB_{n-1}$ coefficients are not constants

Solving Linear Homogeneous Recurrence Relations

- The basic approach is to look for solutions of the form $a_n = r^n$, where r is a constant.
- Note that $a_n = r^n$ is a solution to the recurrence relation $a_n = c_1a_{n-1} + c_2a_{n-2} + \cdots + c_k a_{n-k}$ if and only if $r^n = c_1r^{n-1} + c_2r^{n-2} + \cdots + c_k r^{n-k}$.
- Algebraic manipulation yields the ***characteristic equation***:

$$r^k - c_1r^{k-1} - c_2r^{k-2} - \cdots - c_{k-1}r - c_k = 0$$

Solving Linear Homogeneous Recurrence Relations

- Algebraic manipulation yields the ***characteristic equation***:

  $$r^k - c_1 r^{k-1} - c_2 r^{k-2} - \cdots - c_{k-1} r - c_k = 0$$

- The sequence $\{a_n\}$ with $a_n = r^n$ is a solution if and only if r is a solution to the characteristic equation.
- The solutions to the characteristic equation are called the ***characteristic roots*** of the recurrence relation. The roots are used to give an explicit formula for all the solutions of the recurrence relation.

Solving Linear Homogeneous Recurrence Relations of Degree Two

Theorem 1: Let c_1 and c_2 be real numbers. Suppose that $r^2 - c_1 r - c_2 = 0$ has two distinct roots r_1 and r_2. Then the sequence $\{a_n\}$ is a solution to the recurrence relation $a_n = c_1 a_{n-1} + c_2 a_{n-2}$ if and only if

$$a_n = \alpha r_1^n + \alpha_2 r_2^n$$

for $n = 0, 1, 2, \ldots$, where α_1 and α_2 are constants.

Using Theorem 1

Example: What is the solution to the recurrence relation

$a_n = a_{n-1} + 2a_{n-2}$ with $a_0 = 2$ and $a_1 = 7$?

Solution: The characteristic equation is $r^2 - r - 2 = 0$.

Its roots are $r = 2$ and $r = -1$. Therefore, $\{a_n\}$ is a solution to the recurrence relation if and only if $a_n = \alpha_1 2^n + \alpha_2(-1)^n$, for some constants α_1 and α_2.

To find the constants α_1 and α_2, note that

$a_0 = 2 = \alpha_1 + \alpha_2$ and $a_1 = 7 = \alpha_1 2 + \alpha_2(-1)$.

Solving these equations, we find that $\alpha_1 = 3$ and $\alpha_2 = -1$.

Hence, the solution is the sequence $\{a_n\}$ with $a_n = 3\cdot 2^n - (-1)^n$.

An Explicit Formula for the Fibonacci Numbers

We can use Theorem 1 to find an explicit formula for the Fibonacci numbers. The sequence of Fibonacci numbers satisfies the recurrence relation $f_n = f_{n-1} + f_{n-2}$ with the initial conditions: $f_0 = 0$ and $f_1 = 1$.

Solution: The roots of the characteristic equation $r^2 - r - 1 = 0$ are

$$r_1 = \frac{1+\sqrt{5}}{2}$$

$$r_2 = \frac{1-\sqrt{5}}{2}$$

Fibonacci Numbers (*cont*)

Therefore by Theorem 1

$$f_n = \alpha_1 \left(\frac{1+\sqrt{5}}{2}\right)^n + \alpha_2 \left(\frac{1-\sqrt{5}}{2}\right)^n$$

for some constants α_1 and α_2.

Using the initial conditions $f_0 = 0$ and $f_1 = 1$, we have

$$f_0 = \alpha_1 + \alpha_2 = 0$$

$$f_1 = \alpha_1 \left(\frac{1+\sqrt{5}}{2}\right) + \alpha_2 \left(\frac{1-\sqrt{5}}{2}\right) = 1 \text{ .}$$

Solving, we obtain $\alpha_1 = \frac{1}{\sqrt{5}} \ , \quad \alpha_2 = -\frac{1}{\sqrt{5}}$.

Hence,

$$f_n = \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}} \left(\frac{1-\sqrt{5}}{2}\right)^n$$

The Solution when there is a Repeated Root

Theorem 2: Let c_1 and c_2 be real numbers with $c_2 \neq 0$. Suppose that $r^2 - c_1 r - c_2 = 0$ has one repeated root r_0. Then the sequence $\{a_n\}$ is a solution to the recurrence relation $a_n = c_1 a_{n-1} + c_2 a_{n-2}$ if and only if

$$a_n = \alpha r_0^n + \alpha_2 n r_0^n$$

for $n = 0,1,2,\ldots$, where α_1 and α_2 are constants.

Using Theorem 2

Example: What is the solution to the recurrence relation $a_n = 6a_{n-1} - 9a_{n-2}$ with $a_0 = 1$ and $a_1 = 6$?

Solution: The characteristic equation is $r^2 - 6r + 9 = 0$.

The only root is $r = 3$. Therefore, $\{a_n\}$ is a solution to the recurrence relation if and only if

$$a_n = \alpha_1 3^n + \alpha_2 n(3)^n$$

where α_1 and α_2 are constants. To find the constants α_1 and α_2, note that

$$a_0 = 1 = \alpha_1 \quad \text{and} \quad a_1 = 6 = \alpha_1 \cdot 3 + \alpha_2 \cdot 3.$$

Solving, we find that $\alpha_1 = 1$ and $\alpha_2 = 1$.

Hence,

$$a_n = 3^n + n3^n .$$

Solving Linear Homogeneous Recurrence Relations of Arbitrary Degree

- This theorem can be used to solve linear homogeneous recurrence relations with constant coefficients of any degree when the characteristic equation has distinct roots.

Theorem 3: Let $c_1, c_2, \ldots, c_k$ be real numbers. Suppose that the characteristic equation

$$r^k - c_1 r^{k-1} - \cdots - c_k = 0$$

has k distinct roots $r_1, r_2, \ldots, r_k$. Then a sequence $\{a_n\}$ is a solution of the recurrence relation

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \ldots.. + c_k a_{n-k}$$

if and only if

$$a_n = \alpha_1 r_1^n + \alpha_2 r_2^m + \cdots + \alpha_k r_k^n$$

for $n = 0, 1, 2, \ldots$, where $\alpha_1, \alpha_2, \ldots, \alpha_k$ are constants.

The General Case with Repeated Roots Allowed

Theorem 4: Let $c_1, c_2, \ldots, c_k$ be real numbers. Suppose that the characteristic equation

$$r^k - c_1 r^{k-1} - \cdots - c_k = 0$$

has t distinct roots $r_1, r_2, \ldots, r_t$ with multiplicities $m_1, m_2, \ldots, m_t$, respectively so that $m_i \geq 1$ for $i = 0, 1, 2, \ldots, t$ and $m_1 + m_2 + \ldots + m_t = k$. Then a sequence $\{a_n\}$ is a solution of the recurrence relation

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \ldots + c_k a_{n-k}$$

if and only if

$$\begin{aligned} a_n = &(\alpha_{1,0} + \alpha_{1,1} n + \cdots + \alpha_{1,m_1-1} n^{m_1-1}) r_1^n \\ &+ (\alpha_{2,0} + \alpha_{2,1} n + \cdots + \alpha_{2,m_2-1} n^{m_2-1}) r_2^n \\ &+ \cdots + (\alpha_{t,0} + \alpha_{t,1} n + \cdots + \alpha_{t,m_t-1} n^{m_t-1}) r_t^n \end{aligned}$$

for $n = 0, 1, 2, \ldots$, where $\alpha_{i,j}$ are constants for $1 \leq i \leq t$ and $0 \leq j \leq m_{i-1}$.

Divide-and-Conquer Algorithms and Recurrence Relations

Divide-and-Conquer Algorithmic Paradigm

Definition: A *divide-and-conquer algorithm* works by first ***dividing*** a problem into one or more instances of the same problem of smaller size and then ***conquering*** the problem using the solutions of the smaller problems to find a solution of the original problem.

Examples: Binary search and Merge sort

Divide-and-Conquer Recurrence Relations

- Suppose that a recursive algorithm divides a problem of size n into a subproblems.
- Assume each subproblem is of size n/b.
- Suppose $g(n)$ extra operations are needed in the conquer step.
- Then $f(n)$ represents the number of operations to solve a problem of size n satisfies the following recurrence relation:

$$f(n) = af(n/b) + g(n)$$

- This is called a ***divide-and-conquer recurrence relation***.

Example: Binary Search

- Binary search reduces the search for an element in a sequence of size n to the search in a sequence of size $n/2$. Two comparisons are needed to implement this reduction;
  - one to decide whether to search the upper or lower half of the sequence and
  - the other to determine if the sequence has elements.

- Hence, if $f(n)$ is the number of comparisons required to search for an element in a sequence of size n, then

$$f(n) = f(n/2) + 2$$

  when n is even.

Example: Merge Sort

- The merge sort algorithm splits a list of n (assuming n is even) items to be sorted into two lists with $n/2$ items. It uses fewer than n comparisons to merge the two sorted lists.
- Hence, the number of comparisons required to sort a sequence of size n, is no more than than $M(n)$ where

$$M(n) = 2M(n/2) + n.$$

Example: Fast Multiplication of Integers

- An algorithm for the fast multiplication of two $2n$-bit integers (assuming n is even) first splits each of the $2n$-bit integers into two blocks, each of n bits.
- Suppose that a and b are integers with binary expansions of length $2n$. Let

$$a = (a_{2n-1}a_{2n-2} \dots a_1a_0)_2 \text{ and } b = (b_{2n-1}b_{2n-2} \dots b_1b_0)_2 .$$

- Let $a = 2^nA_1 + A_0$, $b = 2^nB_1 + B_0$, where

$$A_1 = (a_{2n-1} \dots a_{n+1}a_n)_2 , A_0 = (a_{n-1} \dots a_1a_0)_2 ,$$
$$B_1 = (b_{2n-1} \dots b_{n+1}b_n)_2 , B_0 = (b_{n-1} \dots b_1b_0)_2.$$

- The algorithm is based on the fact that ab can be rewritten as:

$$ab = (2^{2n} + 2^n)A_1B_1 + 2^n (A_1 - A_0)(B_0 - B_1) + (2^n + 1)A_0B_0.$$

Example: Fast Multiplication of Integers

- Let $a = 2^n A_1 + A_0$, $b = 2^n B_1 + B_0$, where
  $A_1 = (a_{2n-1} \ldots a_{n+1} a_n)_2$, $A_0 = (a_{n-1} \ldots a_1 a_0)_2$,
  $B_1 = (b_{2n-1} \ldots b_{n+1} b_n)_2$, $B_0 = (b_{n-1} \ldots b_1 b_0)_2$.

- $ab = (2^{2n} + 2^n)A_1B_1 + 2^n (A_1 - A_0)(B_0 - B_1) + (2^n + 1)A_0B_0$.

- This identity shows that the multiplication of two $2n$-bit integers can be carried out using three multiplications of n-bit integers, together with additions, subtractions, and shifts.

- Hence, if $f(n)$ is the total number of operations needed to multiply two n-bit integers, then

  $$f(2n) = 3f(n) + Cn$$

  where Cn represents the total number of bit operations; the additions, subtractions and shifts that are a constant multiple of n-bit operations.

Estimating the Size of Divide-and-Conquer Functions

Theorem 1: Let *f* be an increasing function that satisfies the recurrence relation

$$f(n) = af(n/b) + c$$

whenever *n* is divisible by *b*, where $a \geq 1$, *b* is an integer greater than 1, and *c* is a positive real number. Then

$$f(n) \text{ is } \begin{cases} O(n^{\log_b a}) & \text{if} \quad a > 1 \\ O(\log n) & \text{if} \quad a = 1. \end{cases}$$

Furthermore, when $n = b^k$ and $a \neq 1$, where *k* is a positive integer,

$$f(n) = C_1 n^{\log_b a} + C_2$$

where $C_1 = f(1) + c/(a-1)$ and $C_1 = -c/(a-1)$.

Proof for Theorem 1

Proof: First let $n = b^k$. From the expression for $f(n)$ obtained in the discussion preceding the theorem, with $g(n) = c$, we have

$$f(n) = a^k f(1) + \sum_{j=0}^{k-1} a^j c = a^k f(1) + c \sum_{j=0}^{k-1} a^j.$$

When $a = 1$ we have

$$f(n) = f(1) + ck.$$

Because $n = b^k$, we have $k = \log_b n$. Hence,

$$f(n) = f(1) + c \log_b n.$$

When n is not a power of b, we have $b^k < n < b^{k+1}$, for a positive integer k. Because f is increasing, it follows that $f(n) \le f(b^{k+1}) = f(1) + c(k+1) = (f(1) + c) + ck \le (f(1) + c) + c \log_b n$. Therefore, in both cases, $f(n)$ is $O(\log n)$ when $a = 1$.

Now suppose that $a > 1$. First assume that $n = b^k$, where k is a positive integer. From the formula for the sum of terms of a geometric progression (Theorem 1 in Section 2.4), it follows that

Proof for Theorem 1

Now suppose that $a > 1$. First assume that $n = b^k$, where k is a positive integer. From the formula for the sum of terms of a geometric progression (Theorem 1 in Section 2.4), it follows that

$$\begin{aligned} f(n) &= a^k f(1) + c(a^k - 1)/(a - 1) \\ &= a^k[f(1) + c/(a - 1)] - c/(a - 1) \\ &= C_1 n^{\log_b a} + C_2, \end{aligned}$$

because $a^k = a^{\log_b n} = n^{\log_b a}$ (see Exercise 4 in Appendix 2), where $C_1 = f(1) + c/(a - 1)$ and $C_2 = -c/(a - 1)$.

Now suppose that n is not a power of b. Then $b^k < n < b^{k+1}$, where k is a nonnegative integer. Because f is increasing,

$$\begin{aligned} f(n) &\leq f(b^{k+1}) = C_1 a^{k+1} + C_2 \\ &\leq (C_1 a) a^{\log_b n} + C_2 \\ &= (C_1 a) n^{\log_b a} + C_2, \end{aligned}$$

because $k \leq \log_b n < k + 1$.

Hence, we have $f(n)$ is $O(n^{\log_b a})$. ◁

Complexity of Binary Search

Binary Search Example: Give a big-O estimate for the number of comparisons used by a binary search.

Solution: Since the number of comparisons used by binary search is $f(n) = f(n/2) + 2$ where n is even, by Theorem 1, it follows that $f(n)$ is $O(\log n)$.

Estimating the Size of Divide-and-conquer Functions (*cont*)

Theorem 2. Master Theorem: Let f be an increasing function that satisfies the recurrence relation

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

whenever $n = b^k$, where k is a positive integer greater than 1, and c and d are real numbers with c positive and d nonnegative. Then

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

Complexity of Merge Sort

Merge Sort Example: Give a big-O estimate for the number of comparisons used by merge sort.

Solution: Since the number of comparisons used by merge sort to sort a list of n elements is less than $M(n)$ where $M(n) = 2M(n/2) + n$, by the master theorem $M(n)$ is $O(n \log n)$.

Complexity of Fast Integer Multiplication Algorithm

Integer Multiplication Example: Give a big-O estimate for the number of bit operations used needed to multiply two n-bit integers using the fast multiplication algorithm.

Solution: We have shown that $f(n) = 3f(n/2) + Cn$, when n is even, where $f(n)$ is the number of bit operations needed to multiply two n-bit integers. Hence by the master theorem with $a = 3$, $b = 2$, $c = C$, and $d = 1$ (so that we have the case where $a > b^d$), it follows that $f(n)$ is $O(n^{\log 3})$.

Note that $\log 3 \approx 1.6$. Therefore the fast multiplication algorithm is a substantial improvement over the conventional algorithm that uses $O(n^2)$ bit operations.

Generating Functions

Generating Functions

- Definition 1: The *generation function* for the sequence $a_0, a_1, \ldots, a_k, \ldots$ of real numbers is the infinite series
- $G(x) = a_0 + a_1 x + \ldots + a_k x^k + \ldots = \Sigma_{k=0}^{\infty} a_k x^k$

- Example 1: The generating functions for the sequences $\{a_k\}$ with $a_k=3$, $a_k=k+1$, and $a_k= 2^k$ *are* $\Sigma_{k=0}^{\infty} 3x^k$, $\Sigma_{k=0}^{\infty} (k+1)x^k$, and $\Sigma_{k=0}^{\infty} 2^k x^k$, respectively.

- Example 2: what is the generating function for the sequence 1, 1, 1, 1, 1, 1?

Solution: The generating function of 1, 1, 1, 1, 1, 1 is

$$1 + x + x^2 + x^3 + x^4 + x^5.$$

$$(x^6 - 1)/(x - 1) = 1 + x + x^2 + x^3 + x^4 + x^5$$

Consequently, $G(x) = (x^6 - 1)/(x - 1)$

Useful Facts About Power Series

- Example 3: Let m be a positive integer. Let a_k= C(m,k), for k=0, 1, 2,. . . m,. What is the generating function for the sequence $a_0, a_1, . . ., a_m$?

Solution: The generating function for this sequence is

$$G(x) = C(m, 0) + C(m, 1)x + C(m, 2)x^2 + \cdots + C(m, m)x^m.$$

The binomial theorem shows that $G(x) = (1 + x)^m$.

Useful Facts About Power Series

- Example 4: The function $f(x)=1/(1-x)$ is the generating function of the sequence 1, 1, 1, 1,. . ., because $1/(1-x)= 1 + x + x^2+. . .$. for $|x|<1$.

- Example 5: The function $f(x)=1/(1-ax)$ is the generating function of the sequence $1, a, a^2, a^3, . . .$ because $1/(1-ax)=1 + ax + a^2x^2 + . . .$

when $|ax|<1$, or equivalently, for $|x|<1/|a|$ for $|a|\neq 0$

Useful Facts About Power Series

- Theorem 1: Let f(x)= $\Sigma_{k=0}^{\infty} a_k x^k$ and g(x)= $\Sigma_{k=0}^{\infty} b_k x^k$.

- Then $f(x)+g(x)= \Sigma_{k=0}^{\infty} (a_k + b_k)x^k$ and

- $f(x)g(x) = \Sigma_{k=0}^{\infty} \Sigma_{j=0}^{k} (a_j \times b_{k-j})x^k$

- Example 6: Let $f(x) = 1/(1-x)^2$. Use example 4 to find the coefficients $a_0, a_1, \ldots,$ in the expansion f(x)= $\Sigma_{k=0}^{\infty} a_k x^k$.

$$1/(1-x) = 1 + x + x^2 + x^3 + \cdots$$

$$1/(1-x)^2 = \sum_{k=0}^{\infty}\left(\sum_{j=0}^{k} 1\right) x^k = \sum_{k=0}^{\infty}(k+1)x^k.$$

Useful Facts About Power Series

- Definition 2: Let *u* be a real number and k a nonnegative integer then the *extended binomial coefficient* $\binom{u}{k}$ is defined by
- $$\binom{u}{k} = \begin{cases} u(u-1)\cdots(u-k+1)/k! & \text{if k} > 0 \\ 1 & \text{if k} = 0 \end{cases}$$

- Example 7: Find the values of the extended binomial coefficients $\binom{-2}{3}$ and $\binom{1/2}{3}$.

$$\binom{-2}{3} = \frac{(-2)(-3)(-4)}{3!} = -4.$$

$$\begin{aligned} \binom{1/2}{3} &= \frac{(1/2)(1/2-1)(1/2-2)}{3!} \\ &= (1/2)(-1/2)(-3/2)/6 \\ &= 1/16. \end{aligned}$$

Useful Facts About Power Series

- Example 8: When the top parameter is a negative integer, the extended binomial coefficient can be expressed in terms of an ordinary binomial coefficient . note that

$$\binom{-n}{r} = \frac{(-n)(-n-1)\cdots(-n-r+1)}{r!}$$

by definition of extended binomial coefficient

$$= \frac{(-1)^r n(n+1)\cdots(n+r-1)}{r!}$$

factoring out -1 from each term in the numerator

$$= \frac{(-1)^r (n+r-1)(n+r-2)\cdots n}{r!}$$

by the commutative law for multiplication

$$= \frac{(-1)^r (n+r-1)!}{r!(n-1)!}$$

multiplying both the numerator and denominator by (n-1)!

$$= (-1)^r \binom{n+r-1}{r}$$

by the definition of binomial coefficients

$$= (-1)^r C(n+r-1, r)$$

using alternative notation for binomial coefficients

Useful Facts About Power Series

- Theorem 2: The Extended Binomial Theorem Let *x* be a real number with |x|< 1 and let *u* be a real number. Then

$$(1+x)^u = \sum_{k=0}^{\infty} \binom{u}{k} x^k$$

- Example 9: Find the generating functions for $(1+x)^{-n}$ and $(1-x)^{-n}$, where n is a positive integer, using the extended Binomial Theorem.

$$(1+x)^{-n} = \sum_{k=0}^{\infty} \binom{-n}{k} x^k.$$

$$(1+x)^{-n} = \sum_{k=0}^{\infty} (-1)^k C(n+k-1, k) x^k.$$

$$(1-x)^{-n} = \sum_{k=0}^{\infty} C(n+k-1, k) x^k.$$

Counting Problems and Generating Functions

TABLE 1 **Useful Generating Functions.**

$G(x)$	a_k
$(1+x)^n = \sum_{k=0}^{n} C(n,k)x^k$ $= 1 + C(n,1)x + C(n,2)x^2 + \cdots + x^n$	$C(n,k)$
$(1+ax)^n = \sum_{k=0}^{n} C(n,k)a^k x^k$ $= 1 + C(n,1)ax + C(n,2)a^2x^2 + \cdots + a^n x^n$	$C(n,k)a^k$
$(1+x^r)^n = \sum_{k=0}^{n} C(n,k)x^{rk}$ $= 1 + C(n,1)x^r + C(n,2)x^{2r} + \cdots + x^{rn}$	$C(n,k/r)$ if $r \mid k$; 0 otherwise
$\frac{1-x^{n+1}}{1-x} = \sum_{k=0}^{n} x^k = 1 + x + x^2 + \cdots + x^n$	1 if $k \le n$; 0 otherwise
$\frac{1}{1-x} = \sum_{k=0}^{\infty} x^k = 1 + x + x^2 + \cdots$	1
$\frac{1}{1-ax} = \sum_{k=0}^{\infty} a^k x^k = 1 + ax + a^2x^2 + \cdots$	a^k

Counting Problems and Generating Functions

TABLE 1 Useful Generating Functions.

$G(x)$	a_k
$\frac{1}{1-x^r} = \sum_{k=0}^{\infty} x^{rk} = 1 + x^r + x^{2r} + \cdots$	1 if $r \mid k$; 0 otherwise
$\frac{1}{(1-x)^2} = \sum_{k=0}^{\infty} (k+1)x^k = 1 + 2x + 3x^2 + \cdots$	$k+1$
$\frac{1}{(1-x)^n} = \sum_{k=0}^{\infty} C(n+k-1,k)x^k = 1 + C(n,1)x + C(n+1,2)x^2 + \cdots$	$C(n+k-1,k) = C(n+k-1,n-1)$
$\frac{1}{(1+x)^n} = \sum_{k=0}^{\infty} C(n+k-1,k)(-1)^k x^k = 1 - C(n,1)x + C(n+1,2)x^2 - \cdots$	$(-1)^k C(n+k-1,k) = (-1)^k C(n+k-1,n-1)$
$\frac{1}{(1-ax)^n} = \sum_{k=0}^{\infty} C(n+k-1,k)a^k x^k = 1 + C(n,1)ax + C(n+1,2)a^2x^2 + \cdots$	$C(n+k-1,k)a^k = C(n+k-1,n-1)a^k$
$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$	$1/k!$
$\ln(1+x) = \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k} x^k = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \cdots$	$(-1)^{k+1}/k$

Counting Problems and Generating Functions

- Example 10: Find the number of solutions of

$$e_1 + e_2 + e_3 = 17$$

Where e_1, e_2, e_3 are nonnegative integers with

$2 \le e_1 \le 5$, $3 \le e_2 \le 6$, $4 \le e_3 \le 7$

Solution: The number of solutions with the indicated constraints is the coefficient of x^{17} in the expansion of

$$(x^2 + x^3 + x^4 + x^5)(x^3 + x^4 + x^5 + x^6)(x^4 + x^5 + x^6 + x^7).$$

Counting Problems and Generating Functions

- Example 11: In how many different ways can eight identical cookies be distributed among three distinct children if each child receives at least two cookies and no more than four cookies?

Solution: Because each child receives at least two but no more than four cookies, for each child there is a factor equal to

$$(x^2 + x^3 + x^4)$$

in the generating function for the sequence $\{c_n\}$, where c_n is the number of ways to distribute n cookies. Because there are three children, this generating function is

$$(x^2 + x^3 + x^4)^3.$$

We need the coefficient of x^8 in this product.

Counting Problems and Generating Functions

- Example 12: Use generating functions to determine the number of ways to insert tokens worth \$1, \$2, and \$5 into a vending machine to pay for an item that costs *r* dollars in both the cases then the order in which the tokens are inserted does not matter and when the order does matter.

Solution: Consider the case when the order in which the tokens are inserted does not matter. Here, all we care about is the number of each token used to produce a total of r dollars. Because we can use any number of \$1 tokens, any number of \$2 tokens, and any number of \$5 tokens, the answer is the coefficient of x^r in the generating function

$$(1 + x + x^2 + x^3 + \cdots)(1 + x^2 + x^4 + x^6 + \cdots)(1 + x^5 + x^{10} + x^{15} + \cdots).$$

When the order in which the tokens are inserted matters, the number of ways to insert exactly n tokens to produce a total of r dollars is the coefficient of x^r in

$$(x + x^2 + x^5)^n,$$

Counting Problems and Generating Functions

- Example 13: Use generating functions to find the number of *k*-combinations of a set with n elements. Assume that the Binomial Theorem has already been established.

Solution: Each of the n elements in the set contributes the term $(1 + x)$ to the generating function $f(x) = \sum_{k=0}^{n} a_k x^k$. Here $f(x)$ is the generating function for $\{a_k\}$, where a_k represents the number of k-combinations of a set with n elements. Hence,

$$f(x) = (1 + x)^n.$$

$$f(x) = \sum_{k=0}^{n} \binom{n}{k} x^k,$$

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}.$$

Counting Problems and Generating Functions

- Example 14: Use generating functions to find the number of *r*-combinations from a set with *n* elements when repetition of elements is allowed.

Solution: Let $G(x)$ be the generating function for the sequence $\{a_r\}$, where a_r equals the number of r-combinations of a set with n elements with repetitions allowed. That is, $G(x) = \sum_{r=0}^{\infty} a_r x^r$. Because we can select any number of a particular member of the set with n elements when we form an r-combination with repetition allowed, each of the n elements contributes $(1 + x + x^2 + x^3 + \cdots)$ to a product expansion for $G(x)$. Each element contributes this factor because it may be selected zero times, one time, two times, three times, and so on, when an r-combination is formed (with a total of r elements selected). Because there are n elements in the set and each contributes this same factor to $G(x)$, we have

$$G(x) = (1 + x + x^2 + \cdots)^n.$$

As long as $|x| < 1$, we have $1 + x + x^2 + \cdots = 1/(1 - x)$, so $G(x) = 1/(1 - x)^n = (1 - x)^{-n}$.

$$(1 - x)^{-n} = (1 + (-x))^{-n} = \sum_{r=0}^{\infty} \binom{-n}{r} (-x)^r.$$

$$\begin{aligned} \binom{-n}{r} (-1)^r &= (-1)^r C(n + r - 1, r) \cdot (-1)^r \\ &= C(n + r - 1, r). \end{aligned}$$

Counting Problems and Generating Functions

- Example 15: Use generating functions to find the number of ways to select *r* objects of *n* different kinds if we must select at least one object of each kind.

Solution: Because we need to select at least one object of each kind, each of the n kinds of objects contributes the factor $(x + x^2 + x^3 + \cdots)$ to the generating function $G(x)$ for the sequence $\{a_r\}$, where a_r is the number of ways to select r objects of n different kinds if we need at least one object of each kind. Hence,

$$G(x) = (x + x^2 + x^3 + \cdots)^n = x^n(1 + x + x^2 + \cdots)^n = x^n/(1 - x)^n.$$

$$\begin{aligned}
G(x) &= x^n/(1-x)^n \\
&= x^n \cdot (1-x)^{-n} \\
&= x^n \sum_{r=0}^{\infty} \binom{-n}{r} (-x)^r \\
&= x^n \sum_{r=0}^{\infty} (-1)^r C(n+r-1, r)(-1)^r x^r
\end{aligned}$$

$$\begin{aligned}
&= \sum_{r=0}^{\infty} C(n+r-1, r) x^{n+r} \\
&= \sum_{t=n}^{\infty} C(t-1, t-n) x^t \\
&= \sum_{r=n}^{\infty} C(r-1, r-n) x^r.
\end{aligned}$$

Using Generating Functions to Solve Recurrent Relations

- Example 16: Solve the recurrence relation $a_k=3a_{k-1}$ for k=1, 2, 3,. . . and initial condition $a_0=2$.

Solution: Let $G(x)$ be the generating function for the sequence $\{a_k\}$, that is, $G(x) = \sum_{k=0}^{\infty} a_k x^k$.
First note that

$$xG(x) = \sum_{k=0}^{\infty} a_k x^{k+1} = \sum_{k=1}^{\infty} a_{k-1} x^k.$$

Using the recurrence relation, we see that

$$\begin{aligned} G(x) - 3xG(x) &= \sum_{k=0}^{\infty} a_k x^k - 3\sum_{k=1}^{\infty} a_{k-1} x^k \\ &= a_0 + \sum_{k=1}^{\infty} (a_k - 3a_{k-1}) x^k \\ &= 2, \end{aligned}$$

because $a_0 = 2$ and $a_k = 3a_{k-1}$. Thus,

$$G(x) - 3xG(x) = (1 - 3x)G(x) = 2.$$

Solving for $G(x)$ shows that $G(x) = 2/(1 - 3x)$. Using the identity $1/(1 - ax) = \sum_{k=0}^{\infty} a^k x^k$, from Table 1, we have

$$G(x) = 2\sum_{k=0}^{\infty} 3^k x^k = \sum_{k=0}^{\infty} 2 \cdot 3^k x^k.$$

Consequently, $a_k = 2 \cdot 3^k$.

Using Generating Functions to Solve Recurrent Relations

- Example 17: Suppose that a valid codeword is an n-digit number in decimal notation containing an even number of 0s. Let a_n denote the number of valid codewords of length n . Suppose that the sequence $\{a_n\}$ satisfies the recurrence relation $a_n = 8a_{n-1} + 10^{n-1}$
- And the initial condition $a_1=9$. use generating functions to find an explicit formula for a_n .

Solution: To make our work with generating functions simpler, we extend this sequence by setting $a_0 = 1$; when we assign this value to a_0 and use the recurrence relation, we have $a_1 = 8a_0 + 10^0 = 8 + 1 = 9$, which is consistent with our original initial condition. (It also makes sense because there is one code word of length 0—the empty string.)

We multiply both sides of the recurrence relation by x^n to obtain

$$a_n x^n = 8a_{n-1}x^n + 10^{n-1}x^n.$$

Using Generating Functions to Solve Recurrent Relations

$$a_n x^n = 8a_{n-1}x^n + 10^{n-1}x^n.$$

Let $G(x) = \sum_{n=0}^{\infty} a_n x^n$ be the generating function of the sequence $a_0, a_1, a_2, \ldots$. We sum both sides of the last equation starting with $n = 1$, to find that

$$\begin{aligned}
G(x) - 1 = \sum_{n=1}^{\infty} a_n x^n &= \sum_{n=1}^{\infty} (8a_{n-1}x^n + 10^{n-1}x^n) \\
&= 8\sum_{n=1}^{\infty} a_{n-1}x^n + \sum_{n=1}^{\infty} 10^{n-1}x^n \\
&= 8x\sum_{n=1}^{\infty} a_{n-1}x^{n-1} + x\sum_{n=1}^{\infty} 10^{n-1}x^{n-1} \\
&= 8x\sum_{n=0}^{\infty} a_n x^n + x\sum_{n=0}^{\infty} 10^n x^n \\
&= 8xG(x) + x/(1-10x),
\end{aligned}$$

$$G(x) - 1 = 8xG(x) + x/(1-10x).$$

Solving for $G(x)$ shows that

$$G(x) = \frac{1-9x}{(1-8x)(1-10x)}.$$

Proving Identities via Generating Functions

- Example 18: Use generating functions to show that

$$\Sigma_{k=0}^{n} C(n, k)^2 = C(2n, n)$$

whenever n is a positive integer.

Solution: First note that by the binomial theorem $C(2n, n)$ is the coefficient of x^n in $(1 + x)^{2n}$. However, we also have

$$\begin{aligned}(1 + x)^{2n} &= [(1 + x)^n]^2 \\ &= [C(n, 0) + C(n, 1)x + C(n, 2)x^2 + \cdots + C(n, n)x^n]^2.\end{aligned}$$

The coefficient of x^n in this expression is

$$C(n, 0)C(n, n) + C(n, 1)C(n, n - 1) + C(n, 2)C(n, n - 2) + \cdots + C(n, n)C(n, 0).$$

This equals $\sum_{k=0}^{n} C(n, k)^2$, because $C(n, n - k) = C(n, k)$. Because both $C(2n, n)$ and $\sum_{k=0}^{n} C(n, k)^2$ represent the coefficient of x^n in $(1 + x)^{2n}$, they must be equal. ◀

Next class

- Topic: Relations
- Pre-class reading: Chap 9