Proof Terminology

- A ***proof*** is a valid argument that establishes the truth of a mathematical statement
- ***Axiom*** (or ***postulate***): a statement that is assumed to be true
- ***Theorem***
  - A statement that has been proven to be true
- ***Hypothesis***, ***premise***
  - An assumption (often unproven) defining the structures about which we are reasoning

More Proof Terminology

- ***Lemma***
  - A minor theorem used as a stepping-stone to proving a major theorem.
- ***Corollary***
  - A minor theorem proved as an easy consequence of a major theorem.
- ***Conjecture***
  - A statement whose truth value has not been proven. (A conjecture may be widely believed to be true, regardless.)

Proof Methods

- For proving a statement *p* alone
  - ***Proof by Contradiction*** (indirect proof): Assume $\neg p$, and prove $\neg p \rightarrow \mathbf{F}$.

Proof Methods

- For proving implications $p \rightarrow q$, we have:
  - ***Trivial* proof:** Prove q by itself.
  - ***Direct* proof:** Assume p is true, and prove q.
  - ***Indirect* proof:**
    - ***Proof by Contraposition*** ($\neg q \rightarrow \neg p$): Assume $\neg q$, and prove $\neg p$.
    - ***Proof by Contradiction***: Assume $p \wedge \neg q$, and show this leads to a contradiction. (i.e. prove ($p \wedge \neg q$) $\rightarrow$ **F)**
  - ***Vacuous* proof:** Prove $\neg p$ by itself.

Direct Proof Example

- **<u>Definition</u>:** An integer n is called *odd* iff $n=2k+1$ for some integer k; n is *even* iff $n=2k$ for some k.
- **Theorem:** Every integer is either odd or even, but not both.
  - This can be proven from even simpler axioms.
- **Theorem:**
  (For all integers n) If n is odd, then n^2 is odd.

  <u>Proof</u>:

  If n is odd, then $n = 2k + 1$ for some integer k.

  Thus, $n^2 = (2k + 1)^2 = 4k^2 + 4k + 1 = 2(2k^2 + 2k) + 1$.

  Therefore n^2 is of the form $2j + 1$ (with j the integer $2k^2 + 2k$), thus n^2 is odd. ■

Indirect Proof Example: Proof by Contraposition

- **Theorem:** (For all integers n)
  If $3n + 2$ is odd, then n is odd.
- **Proof:**

  (Contrapositive: If n is even, then $3n + 2$ is even)

  Suppose that the conclusion is false, *i.e.*, that n is even.

  Then $n = 2k$ for some integer k.

  Then $3n + 2 = 3(2k) + 2 = 6k + 2 = 2(3k + 1)$.

  Thus $3n + 2$ is even, because it equals $2j$ for an integer $j = 3k + 1$. So $3n + 2$ is not odd.

  We have shown that $\neg(n \text{ is odd}) \rightarrow \neg(3n + 2 \text{ is odd})$, thus its contrapositive $(3n + 2 \text{ is odd}) \rightarrow (n \text{ is odd})$ is also true. ■

Vacuous Proof Example

- Show $\neg p$ (i.e. p is false) to prove $p \rightarrow q$ is true.

- **Theorem:** (For all n) If n is both odd and even, then $n^2 = n + n$.
- **Proof:**

  The statement "n is both odd and even" is necessarily false, since no number can be both odd and even. So, the theorem is vacuously true. ■

Trivial Proof Example

- Show q (i.e. q is true) to prove $p \to q$ is true.

- **Theorem:** (For integers n) If n is the sum of two prime numbers, then either n is odd or n is even.
- **Proof:**

  Any integer n is either odd or even. So the conclusion of the implication is true regardless of the truth of the hypothesis. Thus the implication is true trivially. ■

Proof by Contradiction

- A method for proving p.
  - Assume $\neg p$, and prove both q and $\neg q$ for some proposition q. (Can be anything!)
  - Thus $\neg p \rightarrow (q \wedge \neg q)$
  - $(q \wedge \neg q)$ is a trivial contradiction, equal to **F**
  - Thus $\neg p \rightarrow$ **F**, which is only true if $\neg p =$ **F**
  - Thus p is true

Rational Number

- Definition:

  The real number r is *rational* if there exist integers p and q with $q \neq 0$ such that $r = p/q$. A real number that is not rational is called *irrational*.

Proof by Contradiction Example

- **Theorem:** $\sqrt{2}$ is irrational.
  - **Proof:**
  - Assume that $\sqrt{2}$ is rational. This means there are integers x and y ($y \neq 0$) with no common divisors such that $\sqrt{2} = x/y$.

    Squaring both sides, $2 = x^2/y^2$, so $2y^2 = x^2$. So x^2 is even; thus x is even (see earlier).

    Let $x = 2k$. So $2y^2 = (2k)^2 = 4k^2$. Dividing both sides by 2, $y^2 = 2k^2$. Thus y^2 is even, so y is even.

    But then x and y have a common divisor, namely 2, so we have a contradiction.

    Therefore, $\sqrt{2}$ is irrational. ■

Proof by Contradiction

- Proving implication $p \rightarrow q$ by contradiction
  - Assume $\neg q$, and use the premise p to arrive at a contradiction, i.e. $(\neg q \wedge p) \rightarrow \mathbf{F}$ $(p \rightarrow q \equiv (\neg q \wedge p) \rightarrow \mathbf{F})$
  - How does this relate to the proof by contraposition?
  - ***Proof by Contraposition*** $(\neg q \rightarrow \neg p)$: Assume $\neg q$, and prove $\neg p$.

Proof by Contradiction Example: Implication

- **Theorem:** (For all integers n)

  If $3n + 2$ is odd, then n is odd.

- **Proof:**

  Assume that the conclusion is false, *i.e.*, that n is even, and that $3n + 2$ is odd.

  Then $n = 2k$ for some integer k and $3n + 2 = 3(2k) + 2 = 6k + 2 = 2(3k + 1)$. Thus $3n + 2$ is even, because it equals $2j$ for an integer $j = 3k + 1$.

  This contradicts the assumption "$3n + 2$ is odd".

  This completes the proof by contradiction, proving that if $3n + 2$ is odd, then n is odd. ■

Circular Reasoning

- The fallacy of (explicitly or implicitly) assuming the very statement you are trying to prove in the course of its proof. Example:
- Prove that an integer n is even, if n^2 is even.
- **Attempted proof:**

  Assume n^2 is even. Then $n^2 = 2k$ for some integer k.

  Dividing both sides by n gives $n = (2k)/n = 2(k/n)$.

  So there is an integer j (namely k/n) such that $n = 2j$. Therefore n is even.

  - Circular reasoning is used in this proof. Where?

*Begs the question: How do you show that $j = k/n = n/2$ is an integer, without **first** assumig that n is even?*

The Identity Function

- For any domain A, the ***identity function*** $I: A \to A$ (also written as I_A, **1**, $\mathbf{1}_A$) is the unique function such that $\forall a \in A$: $I(a) = a$.
- Note that the identity function is always both one-to-one and onto (i.e., bijective).
- For a bijection $f : A \to B$ and its inverse function $f^{-1}: B \to A$,

$$f^{-1} \circ f = I_A$$

- Some identity functions you've seen:
  - $+$ 0, $\times$ 1, $\wedge$ **T**, $\vee$ **F**, $\cup\ \varnothing$, $\cap\ U$.

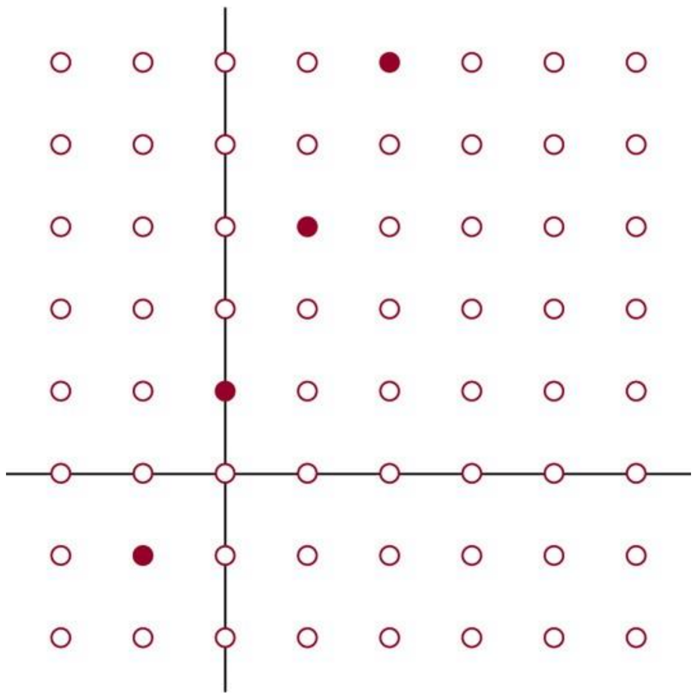
Graphs of Functions

- We can represent a function $f : A \rightarrow B$ as a set of ordered pairs $\{(a, f(a)) \mid a \in A\}$. ← The function's *graph*.

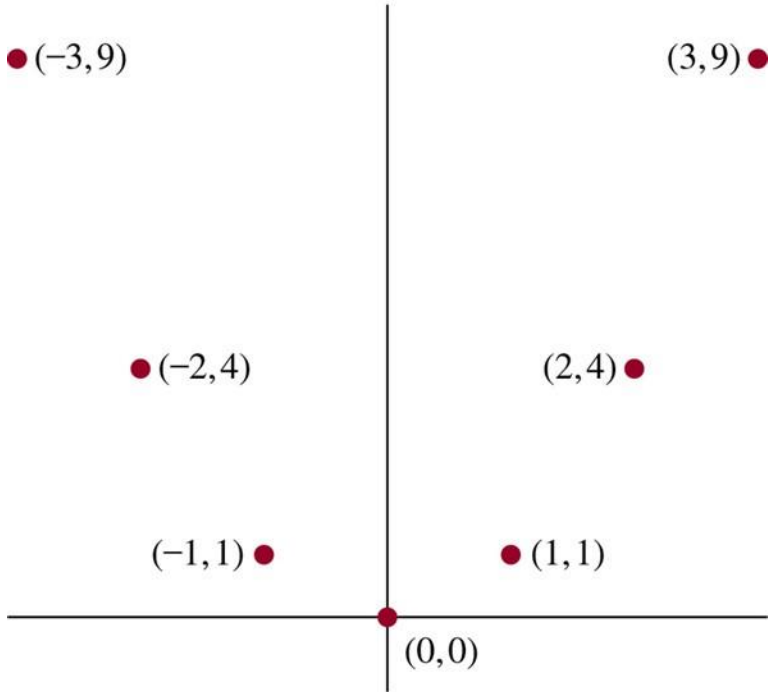
- Note that $\forall a \in A$, there is only 1 pair (a, b).

- For functions over numbers, we can represent an ordered pair (x, y) as a point on a plane.
- A function is then drawn as a curve (set of points), with only one y for each x.

Graphs of Functions: Examples



The graph of $f(n) = 2n + 1$ from **Z** to **Z**



The graph of $f(x) = x^2$ from **Z** to **Z**

Floor&Ceiling Functions

In discrete math, we frequently use the following two functions over real numbers:

- The ***floor function*** $\lfloor\cdot\rfloor$***: R → Z, where*** $\lfloor x \rfloor$ (“**floor** of x”) *means the* ***largest integer*** **≤ *x*,** i.e., $\lfloor x \rfloor = \max(\{i \in \mathbf{Z} \mid i \le x\})$.

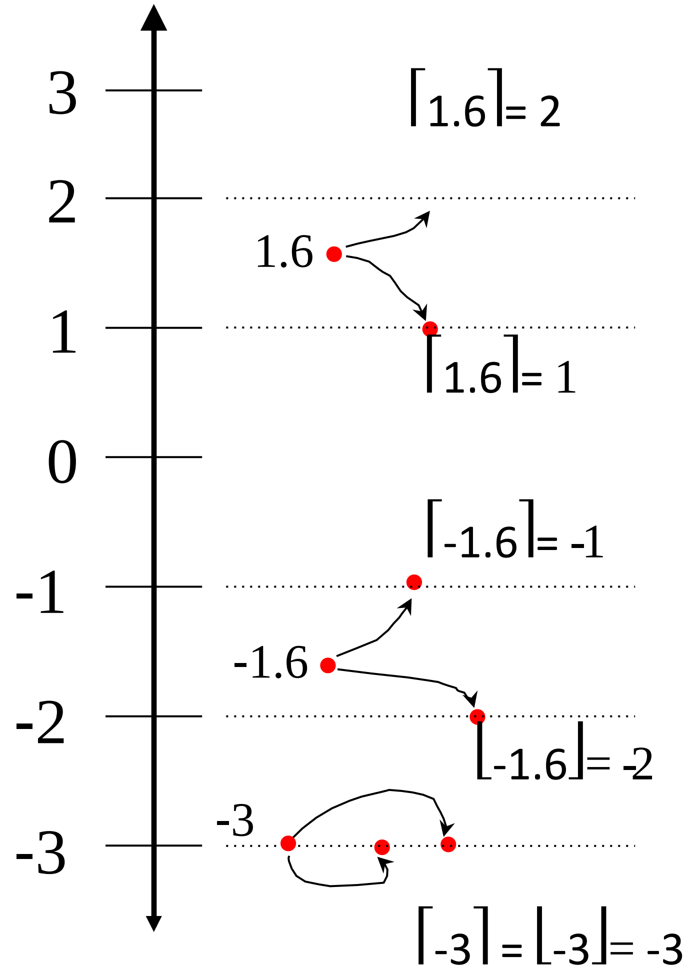
E.g. $\lfloor 2.3 \rfloor = 2$, $\lfloor 5 \rfloor = 5$, $\lfloor -1.2 \rfloor = -2$

- The ***ceiling function*** $\lceil\cdot\rceil$***: R → Z, where*** $\lceil x \rceil$ (“ceiling of x”) *means the* ***smallest integer*** **≥ *x*,** i.e., $\lceil x \rceil = \min(\{i \in \mathbf{Z} \mid i \ge x\})$

E.g. $\lceil 2.3 \rceil = 3$, $\lceil 5 \rceil = 5$, $\lceil -1.2 \rceil = -$

Visualizing Floor & Ceiling

- Real numbers "fall to their floor" or "rise to their ceiling."



- Note that if $x \notin \mathbf{Z}$,

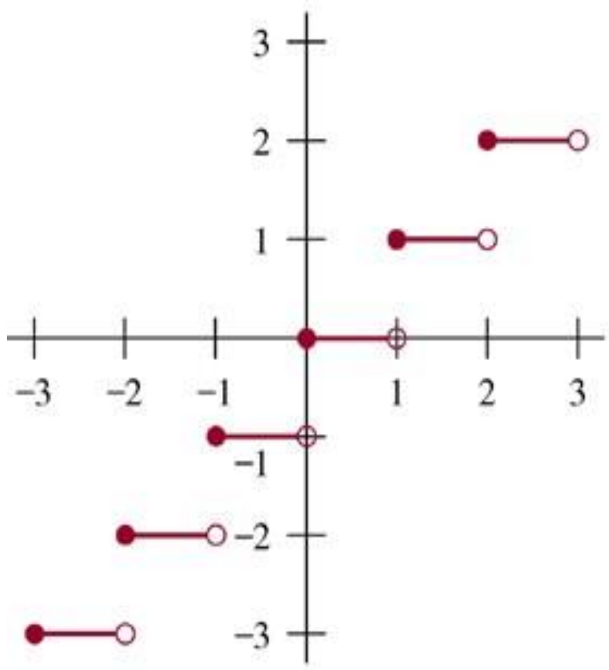
$$\lfloor -x \rfloor \neq -\lfloor x \rfloor \;\&$$

$$\lceil -x \rceil \neq -\lceil x \rceil$$

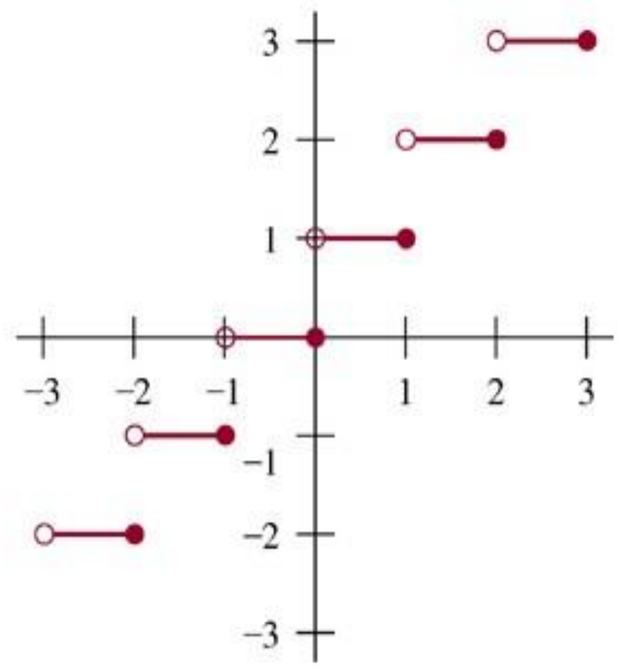
- Note that if $x \in \mathbf{Z}$,

$$\lfloor x \rfloor = \lceil x \rceil = x.$$

Plots with Floor/Ceiling: Example



$y = \lfloor x \rfloor$



$y = \lceil x \rceil$

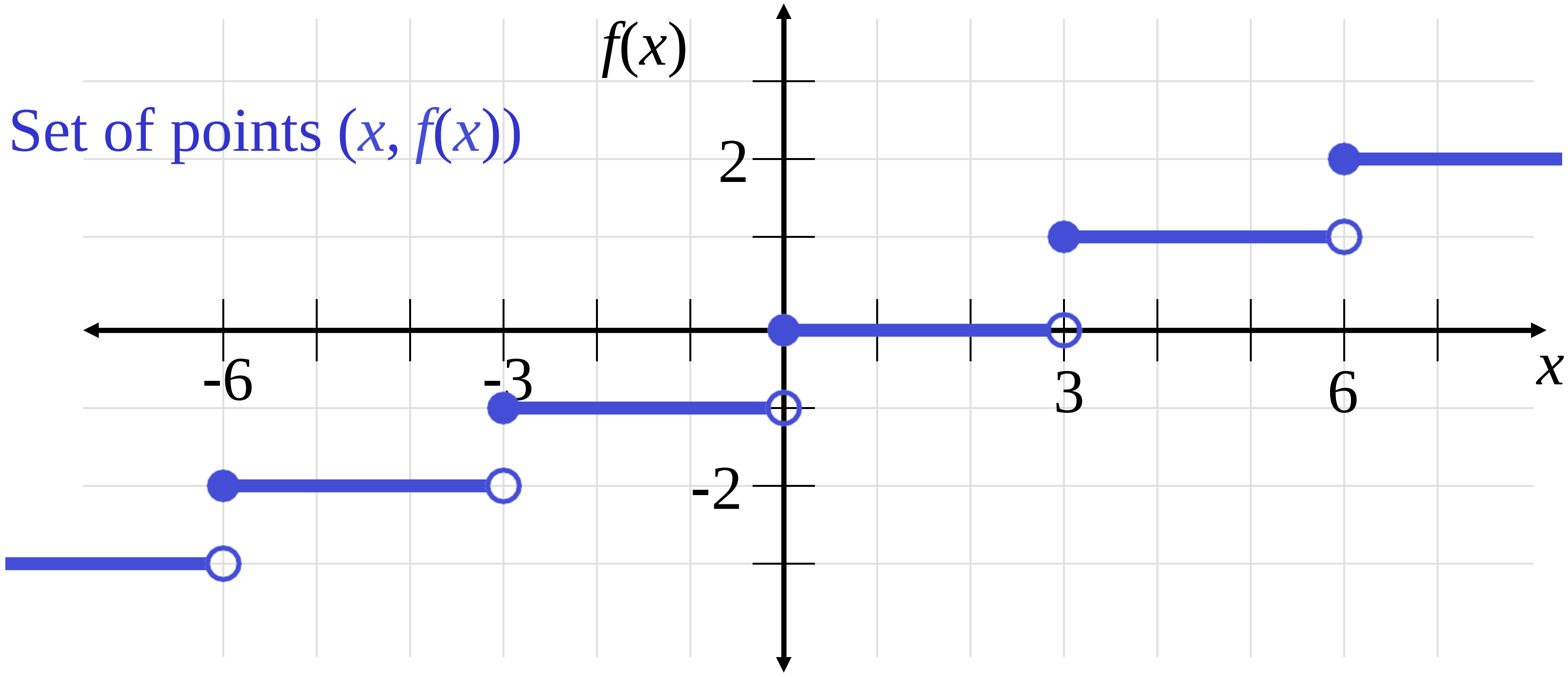
Plots with Floor/Ceiling

- Note that for $f(x) = \leq\lfloor x f \rfloor$, the graph of *f* includes the point (*a*, 0) for all values of *a* such that $0 \leq a < 1$, but not for the value $a = 1$.
- We say that the set of points (*a*, 0) that is in *f* does not include its *limit* or *boundary* point (*a*,1).
- Sets that do not include all of their limit points are called *open sets*.
- In a plot, we draw a limit point of a curve using an open dot (circle) if the limit point is not on the curve, and with a closed (solid) dot if it is on the curve.

Plots with Floor/Ceiling: Another Example

- Plot of graph of function $f(x) = \lfloor x/3 \rfloor$:

Set of points $(x, f(x))$



2.4 Sequences and Summations

- Sequences are ordered lists of elements, represents solutions to certain counting problems.
- A sequence is a discrete structure used to represent an ordered list.
- A *sequence is a function from a subset of the set of integers.*
- *Notation* a_n *to denote the image of the integer n.* a_n *a term of the sequence.*
- A ***summation*** is a compact notation for the sum of the terms in a (possibly infinite) sequence.

Sequences

- A ***sequence*** or ***series*** $\{a_n\}$ is identified with a ***generating function*** $f : I \rightarrow S$ for some subset $I \subseteq \mathbf{N}$ and for some set S.
- Often we have $I = \mathbf{N}$ or $I = \mathbf{Z}^+ = \mathbf{N} - \{0\}$.
- If f is a generating function for a sequence $\{a_n\}$, then for $n \in I$, the symbol a_n denotes $f(n)$, also called ***term*** n of the sequence.
- The ***index*** of a_n is n. (Or, often i is used.)
- A sequence is sometimes denoted by listing its first and/or last few elements, and using ellipsis (…) notation.
- *E.g.*, "$\{a_n\} = 0, 1, 4, 9, 16, 25, \ldots$" is taken to mean $\forall n \in \mathbf{N}, a_n = n^2$.

Sequence Examples

- Some authors write "the sequence $a_1, a_2, \ldots$" instead of $\{a_n\}$, to ensure that the set of indices is clear.
- Be careful: Our book often leaves the indices ambiguous.
- An example of an infinite sequence:
- Consider the sequence $\{a_n\} = a_1, a_2, \ldots,$ where $(\forall n \geq 1)\ a_n = f(n) = 1/n$.
- Then, we have $\{a_n\} = 1, 1/2, 1/3, \ldots$
- Called "harmonic series"

Example with Repetitions

- Like tuples, but unlike sets, a sequence may contain *repeated* instances of an element.
- Consider the sequence $\{b_n\} = b_0, b_1, \ldots$ (note that 0 is an index) where $b_n = (-1)^n$.
- Thus, $\{b_n\} = 1, -1, 1, -1, \ldots$
- Note repetitions!
- This $\{b_n\}$ denotes an infinite sequence of 1's and -1's, <u>*not*</u> the 2-element set {1, -1}.

Geometric Progression

- A geometric progression is a sequence of the form

***a*, *ar*, *ar*^{2}, …, *ar*^{n},…**

where the ***initial term a*** and the ***common ratio r*** are real numbers.

- A geometric progression is a discrete analogue of the exponential function $f(x) = ar^x$
- Examples

Assuming $n = 0, 1, 2, \ldots$

  - $\{b_n\}$ with $b_n = (-1)^n$ — initial term 1, common ratio –1
  - $\{c_n\}$ with $c_n = 2 \cdot 5^n$ — initial term 2, common ratio 5
  - $\{d_n\}$ with $d_n = 6 \cdot (1/3)^n$ — initial term 6, common ratio 1/3

Arithmetic Progression

- An arithmetic progression is a sequence of the form

***a*, *a*+*d*, *a*+2*d*, …, *a*+*nd*,…**

where the ***initial term a*** and the ***common difference d*** are real numbers.

- An arithmetic progression is a discrete analogue of the linear function $f(x) = a + dx$

- Examples

Assuming $n = 0, 1, 2, \ldots$

  - $\{s_n\}$ with $s_n = -1 + 4n$ initial term –1, common diff. 4
  - $\{t_n\}$ with $t_n = 7 - 3n$ initial term 7, common diff. –3

Recognizing Sequences (I)

- Sometimes, you're given the first few terms of a sequence,
- and you are asked to find the sequence's generating function,
- or a procedure to enumerate the sequence.

- Examples: What's the next number?
  - 1, 2, 3, 4,… 5 (the 5th smallest number > 0)
  - 1, 3, 5, 7, 9,… 11 (the 6th smallest odd number > 0)
  - 2, 3, 5, 7, 11,... 13 (the 6th smallest prime number)

Recognizing Sequences (II)

- General problems
  - Given a sequence, find a formula or a general rule that produced it

- Examples: How can we produce the terms of a sequence if the first 10 terms are
  - 1, 2, 2, 3, 3, 3, 4, 4, 4, 4?

Possible match: next five terms would all be 5, the following six terms would all be 6, and so on.

  - 5, 11, 17, 23, 29, 35, 41, 47, 53, 59?

Possible match: nth term is $5 + 6(n - 1) = 6n - 1$ (assuming $n = 1, 2, 3, \ldots$)

Special Integer Sequences

- A useful technique for finding a rule for generating the terms of a sequence is to compare the terms of a sequence of interest with the terms of a well-known integer sequences (e.g. arithmetic/geometric progressions, perfect squares, perfect cubes, etc.)

TABLE 1 Some Useful Sequences.

nth Term	*First 10 Terms*
n^2	1, 4, 9, 16, 25, 36, 49, 64, 81, 100, … .
n^3	1, 8, 27, 64, 125, 216, 343, 512, 729, 1000, … .
n^4	1, 16, 81, 256, 625, 1296, 2401, 4096, 6561, 10000, … .
2^n	2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, … .
3^n	3, 9, 27, 81, 243, 729, 2187, 6561, 19683, 59049, … .
$n!$	1, 2, 6, 24, 120, 720, 5040, 40320, 362880, 3628800, … .

Coding: Fibbonaci Series

- Series $\{a_n\} = \{1, 1, 2, 3, 5, 8, 13, 21, \ldots\}$

- Generating function (recursive definition!):
  - $a_0 = a_1 = 1$ and
  - $a_n = a_{n-1} + a_{n-2}$ for all $n > 1$

- Now let's find the entire series $\{a_n\}$:

```
int [] a = new int [n];
  a[0] = 1;
a[1] =   1;
  for (int i = 2; i < n; i++) {
    a[i] = a[i-1] + a[i-2];
  }
  return a;
```

Coding: Factorial Series

- Factorial series $\{a_n\} = \{1, 2, 6, 24, 120, \ldots\}$

- Generating function:
- $a_n = n! = 1 \times 2 \times 3 \times \ldots \times n$

- This time, let's just find the term a_n:

```
int an = 1;
for (int      i       = 1; i <= n; i++) {
an = an * i;
}
return an;
```

Summation Notation

- Given a sequence $\{a_n\}$, an integer *lower bound (or limit)* $j \geq 0$, and an integer *upper bound* $k \geq j$, then the *summation of* $\{a_n\}$ *from* a_j *to* a_k is written and defined as follows:

$$\boxed{\sum_{i=j}^{k} a_i = a_j + a_{j+1} + \ldots + a_k}$$

- Here, i is called the *index of summation*.

$$\sum_{i=j}^{k} a_i = \sum_{m=j}^{k} a_m = \sum_{l=j}^{k} a_l$$

Generalized Summations

- For an infinite sequence, we write:

$$\sum_{i=j}^{\infty} a_i = a_j + a_{j+1} + \cdots$$

- To sum a function over all members of a set $X = \{x_1, x_2, \ldots\}$:

$$\sum_{x \in X} f(x) = f(x_1) + f(x_2) + \cdots$$

- Or, if $X = \{x \mid P(x)\}$, we may just write:

$$\sum_{P(x)} f(x) = f(x_1) + f(x_2) + \cdots$$

Simple Summation Example

- $$\sum_{i=2}^{4}\left(i^2+1\right)=$$

- $$1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}\cdots+\frac{1}{100}=\sum_{i=1}^{100}\frac{1}{i}$$

More Summation Examples

- An infinite sequence with a finite sum:

$$\sum_{i=0}^{\infty} 2^{-i} = 2^0 + 2^{-1} + \cdots = 1 + \frac{1}{2} + \frac{1}{4} + \cdots = 2$$

- Using a predicate to define a set of elements to sum over:

$$\sum_{\substack{(x \text{ is prime}) \wedge \\ x<10}} x^2 = 2^2 + 3^2 + 5^2 + 7^2$$

$$= 4 + 9 + 25 + 49 = 87$$

Summation Manipulations

- Some handy identities for summations:
  - Summing constant value

$$\sum_{n=i}^{j} c = (j - i + 1) \cdot c$$

Number of terms in the summation

$$\sum_{n=1}^{3} 2 =$$

$$= 6$$

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

$$= 4 \oplus (2i) = 8i$$

Summation Manipulations

- Distributive law

$$\sum_{n=i}^{j} cf(n) = c\sum_{n=i}^{j} f(n)$$

$$\sum_{n=1}^{3}\left(4\cdot n^2\right) = 4\cdot 1^2 + 4\cdot 2^2 + 4\cdot 3^2$$

$$= 4\cdot\left(1^2 + 2^2 + 3^2\right)$$

$$= 4\sum_{n=1}^{3} n^2$$

Summation Manipulations

- An application of commutativity

$$\sum_{n=i}^{j}\left(f(n)+g(n)\right)=\sum_{n=i}^{j}f(n)+\sum_{n=i}^{j}g(n)$$

$$\sum_{n=2}^{4}\left(n+2n\right)=(2+2\cdot 2)+(3+2\cdot 3)+(4+2\cdot 4)$$

$$=(2+3+4)+(2\cdot 2+2\cdot 3+2\cdot 4)$$

$$=\sum_{n=2}^{4}n+\sum_{n=2}^{4}2n$$

Index Shifting

$$\sum_{i=j}^{m} f(i) = \sum_{k=j+n}^{m+n} f(k-n)$$

$$\sum_{i=1}^{4} i^2 = 1^2 + 2^2 + 3^2 + 4^2$$

- Let $k = i + 2$, then $i = k - 2$

$$\sum_{k=1+2}^{4+2} (k-2)^2 = \sum_{k=3}^{6} (k-2)^2$$

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

More Summation Manipulations

- Sequence splitting

$$\sum_{i=j}^{k} f(i) = \sum_{i=j}^{m} f(i) + \sum_{i=m+1}^{k} f(i) \quad \text{if } j \le m < k$$

$$\sum_{i=0}^{4} i^3 = 0^3 + 1^3 + 2^3 + 3^3 + 4^3$$

$$= (0^3 + 1^3 + 2^3) + (3^3 + 4^3)$$

$$= \sum_{i=0}^{2} i^3 + \sum_{i=3}^{4} i^3$$

More Summation Manipulations

- Order reversal

$$\sum_{i=0}^{k} f(i) = \sum_{i=0}^{k} f(k-i)$$

$$\sum_{i=0}^{3} i^3 = 0^3 + 1^3 + 2^3 + 3^3$$

$$= (3-0)^3 + (3-1)^3 + (3-2)^3 + (3-3)^3$$

$$= \sum_{i=0}^{3} (3-i)^3$$

Example: Geometric Progression

- A *geometric progression* is a sequence of the form $a, ar, ar^2, ar^3, \ldots, ar^n, \ldots$ where $a, r \in \mathbf{R}$.
- The sum of such a sequence is given by:

$$\boxed{S = \sum_{i=0}^{n} ar^{i}}$$

- We can reduce this to *closed form* via clever manipulation of summations...

THEOREM 1 If a and r are real numbers and $r \neq 0$, then

$$\sum_{j=0}^{n} ar^j = \begin{cases} \dfrac{ar^{n+1} - a}{r - 1} & \text{if } r \neq 1 \\ (n+1)a & \text{if } r = 1. \end{cases}$$

Proof: Let

$$S_n = \sum_{j=0}^{n} ar^j.$$

To compute S, first multiply both sides of the equality by r and then manipulate the resulti
sum as follows:

$$
\begin{aligned}
r S_n &= r \sum_{j=0}^{n} ar^j && \text{substituting summation formula for } S \\
&= \sum_{j=0}^{n} ar^{j+1} && \text{by the distributive property} \\
&= \sum_{k=1}^{n+1} ar^k && \text{shifting the index of summation, with } k = j+1 \\
&= \left(\sum_{k=0}^{n} ar^k \right) + (ar^{n+1} - a) && \text{removing } k = n+1 \text{ term and adding } k = 0 \text{ term} \\
&= S_n + (ar^{n+1} - a) && \text{substituting } S \text{ for summation formula}
\end{aligned}
$$

From these equalities, we see that

$$rS_n = S_n + (ar^{n+1} - a).$$

Solving for S_n shows that if $r \neq 1$, then

$$S_n = \frac{ar^{n+1} - a}{r - 1}.$$

If $r = 1$, then the $S_n = \sum_{j=0}^{n} ar^j = \sum_{j=0}^{n} a = (n + 1)a.$

Gauss' Trick, Illustrated

- Consider the sum:

$$1+2+\ldots+(n/2)+((n/2)+1)+\ldots+(n-1)+n$$

$n+1$

$\vdots$

$n+1$ $n+1$

- We have $n/2$ pairs of elements, each pair summing to $n+1$, for a total of $(n/2)(n+1)$.

Some Shortcut Expressions

TABLE 2 Some Useful Summation Formulae.

Sum	*Closed Form*	
$\sum_{k=0}^{n} ar^k \ (r \neq 0)$	$\frac{ar^{n+1} - a}{r - 1}, r \neq 1$	Geometric sequence
$\sum_{k=1}^{n} k$	$\frac{n(n+1)}{2}$	Gauss' trick
$\sum_{k=1}^{n} k^2$	$\frac{n(n+1)(2n+1)}{6}$	Quadratic series
$\sum_{k=1}^{n} k^3$	$\frac{n^2(n+1)^2}{4}$	Cubic series
$\sum_{k=0}^{\infty} x^k, \lvert x \rvert < 1$	$\frac{1}{1-x}$	
$\sum_{k=1}^{\infty}, kx^{k-1}, \lvert x \rvert < 1$	$\frac{1}{(1-x)^2}$	

Using the Shortcuts

- Example: Evaluate $\displaystyle\sum_{k=50}^{100} k^2$
- Use series splitting. $\displaystyle\sum_{k=1}^{100} k^2 = \sum_{k=1}^{49} k^2 + \sum_{k=50}^{100} k^2$
- Solve for desired summation. $\displaystyle\sum_{k=50}^{100} k^2 = \sum_{k=1}^{100} k^2 - \sum_{k=1}^{49} k^2$
- Apply quadratic series rule. $\displaystyle = \frac{100\cdot 101\cdot 201}{6} - \frac{49\cdot 50\cdot 99}{6}$
- Evaluate. $= 338{,}350 - 40{,}425$

$= 297{,}925.$

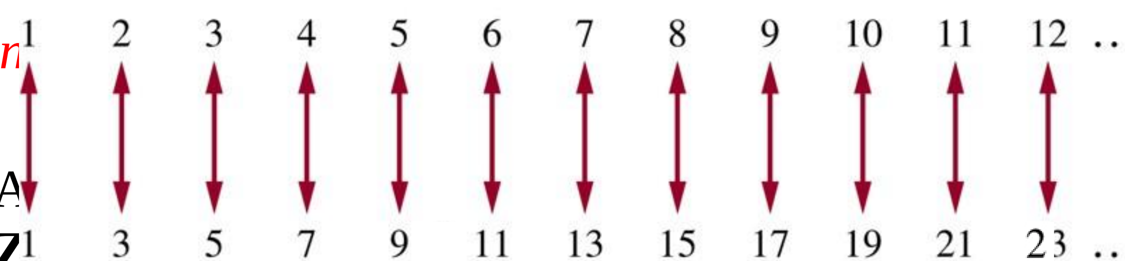
Cardinality

- The sets *A* and *B* have the same ***cardinality*** if and only if there is a one-to-one correspondence from *A* to *B*.
- A set that is either finite or has the same cardinality as the set of positive integers is called **countable**.
- A set that is not countable is called **uncountable**.
- Example: Show that the set of odd positive integers is a countable set.

Consider the function *f*(*n*
integers

A

between **Z**



Summation Manipulations

- Useful identities:

$$\sum_{i=j}^{k} f(i) = \sum_{i=j}^{m} f(i) + \sum_{i=m+1}^{k} f(i) \quad \text{if } j \le m < k$$

(Sequence splitting.)

$$\sum_{i=0}^{k} f(i) = \sum_{i=0}^{k} f(k-i)$$

(Order reversal.)

$$\sum_{i=1}^{2k} f(i) = \sum_{i=1}^{k} \big(f(2i-1) + f(2i) \big)$$

(Grouping.)

Lecture 16

Chapter 4. Induction and Recursion

1. Mathematical Induction
2. Strong Induction

Mathematical Induction

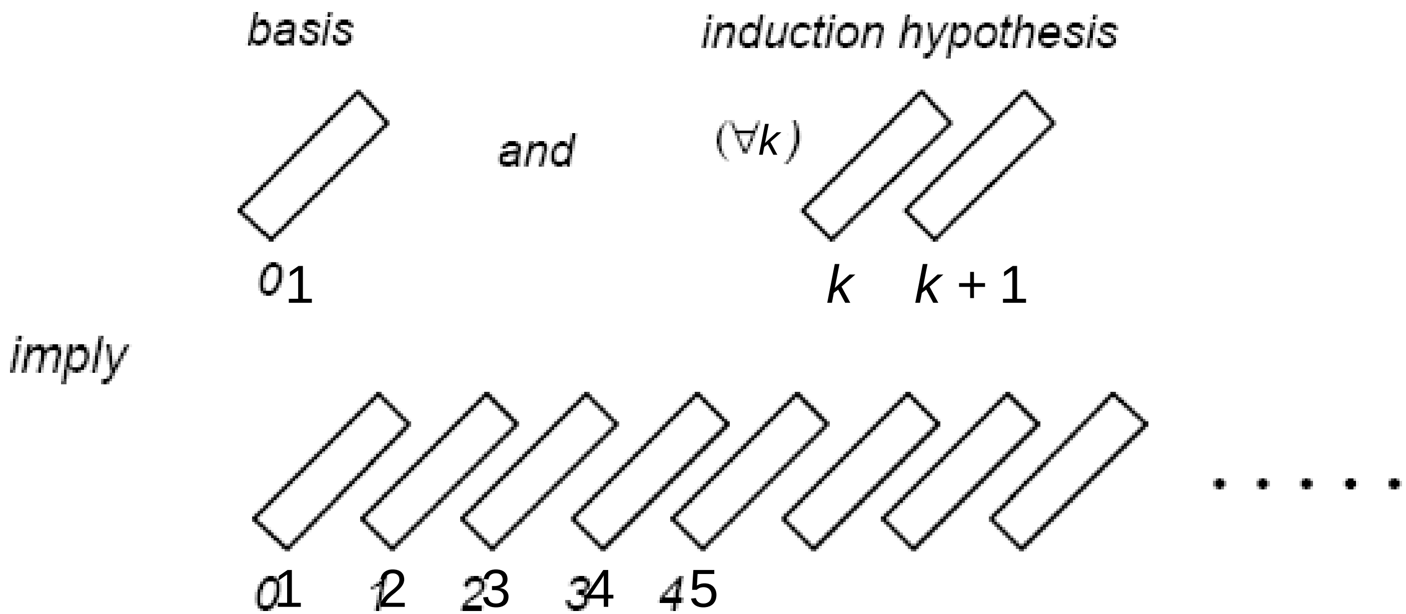
- A powerful, rigorous technique for proving that a statement $P(n)$ is true for ***every*** positive integers n, no matter how large.
- Essentially a "domino effect" principle.
- Based on a predicate-logic inference rule:

$$\begin{array}{l} P(1) \\ \forall k \geq 1\ [P(k) \rightarrow P(k+1)] \\ \hline \therefore \forall n \geq 1\ P(n) \end{array}$$

"The First Principle of Mathematical Induction"

The "Domino Effect"

- **Premise #1:** Domino #1 falls.
- **Premise #2:** For every $k \in \mathbf{Z}^+$, if domino #k falls, then so does domino #k+1.
- **Conclusion:** All of the dominoes fall down!



Note: this works even if there are infinitely many dominoes!

Mathematical Induction

- **PRINCIPLE OF MATHEMATICAL INDUCTION:**

To prove that a statement $P(n)$ is true for all positive integers n, we complete two steps:

- **BASIS STEP**: Verify that $P(1)$ is true

- **INDUCTIVE STEP**: Show that the conditional statement $P(k) \rightarrow P(k+1)$ is true for all positive integers k

Inductive Hypothesis

Validity of Induction

Proof: that $\forall n \geq 1\ P(n)$ is a valid consequent: Given any $k \geq 1$, the 2nd premise $\forall k \geq 1\ (P(k) \rightarrow P(k+1))$ trivially implies that $(P(1) \rightarrow P(2)) \wedge (P(2) \rightarrow P(3)) \wedge \ldots \wedge (P(n-1) \rightarrow P(n))$.

Repeatedly applying the hypothetical syllogism rule to adjacent implications in this list $n - 1$ times then gives us $P(1) \rightarrow P(n)$; which together with $P(1)$ (premise #1) and *modus ponens* gives us $P(n)$.

Thus $\forall n \geq 1\ P(n)$. ■

Outline of an Inductive Proof

- Let us say we want to prove $\forall n \in \mathbf{Z}^+ P(n)$.
  - Do the ***base case*** (or ***basis step***): Prove $P(1)$.
  - Do the ***inductive step***: Prove $\forall k \in \mathbf{Z}^+ P(k) \rightarrow P(k+1)$.
    - *E.g.* you could use a direct proof, as follows:
    - Let $k \in \mathbf{Z}^+$, assume $P(k)$. (*inductive hypothesis*)
    - Now, under this assumption, prove $P(k+1)$.
  - The inductive inference rule then gives us $\forall n \in \mathbf{Z}^+ P(n)$.

Induction Example

- Show that, for $n \geq 1$

$$\boxed{1 + 2 + \cdots + n = \frac{n(n+1)}{2}}$$

- Proof by induction
- $P(n)$: the sum of the first n positive integers is $n(n+1)/2$, i.e. $P(n)$ is
- **_Basis step_**: Let n = 1. The sum of the first positive integer is 1, i.e. $P(1)$ is true.

$$1 = \frac{1(1+1)}{2}$$

Example *(cont.)*

- ***Inductive step***: Prove $\forall k\geq 1$: $P(k)\rightarrow P(k+1)$.
  - Inductive Hypothesis, $P(k)$:

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

  - Let $k\geq 1$, assume $P(k)$, and prove $P(k+1)$, i.e.

$$1+2+\cdots+k+(k+1)=\frac{(k+1)[(k+1)+1]}{2}$$

$$\underbrace{=}_{P(k+1)}\frac{(k+1)(k+2)}{2}$$

Example *(cont.)*

By inductive hypothesis $P(k)$

- ***Inductive step*** continues…

$$
\begin{aligned}
1+2+\cdots+k+(k+1) &= \frac{k(k+1)}{2}+(k+1) \\
&= \frac{k(k+1)}{2}+\frac{2(k+1)}{2} \\
&= \frac{k^2+3k+2}{2} \\
&= \frac{(k+1)(k+2)}{2}
\end{aligned}
$$

$P(k+1)$

- Therefore, by the principle of mathematical induction $P(n)$ is true for all integers n with $n \geq 1$

Induction Example 3

- Prove that $\forall n \geq 1, n < 2^n$. Let $P(n) = (n < 2^n)$
- ***Basis step***: $P(1)$: $(1 < 2^1) \equiv (1 < 2)$: **True**.
- ***Inductive step***: For $k \geq 1$, prove $P(k) \rightarrow P(k+1)$.
- Assuming $k < 2^k$, prove $k + 1 < 2^{k+1}$.
- Note $k + 1 < 2^k + 1$ (by inductive hypothesis)

$< 2^k + 2^k$ (because $1 < 2^k$ for $k \geq 1$)

$= 2 \cdot 2^k = 2^{k+1}$

- So $k + 1 < 2^{k+1}$, i.e. $P(k+1)$ is true
- Therefore, by the principle of mathematical induction $P(n)$ is true for all integers n with $n \geq 1$.

Generalizing Induction

- Rule can also be used to prove $\forall n \geq c\ P(n)$ for a given constant $c \in \mathbf{Z}$, where maybe $c \neq 1$.

- In this circumstance, the basis step is to prove $P(c)$ rather than $P(1)$, and the inductive step is to prove $\forall k \geq c\ (P(k) \rightarrow P(k+1))$.

Induction Example 4

- **Example 6**: Prove that $2^n < n!$ for $n \geq 4$ using mathematical induction.
- $P(n)$: $2^n < n!$
- ***Basis step***: Show that $P(4)$ is true
- Since $2^4 = 16 < 4! = 24$, $P(4)$ is true
- ***Inductive step***: Show that $P(k) \rightarrow P(k+1)$ for $k \geq 4$

$P(k+1)$ is true

$$
\begin{aligned}
2^{k+1} &= 2 \cdot 2^k && \text{(by definition of exponent)} \\
&< 2 \cdot k! && \text{(by the inductive hypothesis } P(k)\text{)} \\
&< (k+1) \cdot k! && \text{(because } 2 < k+1 \text{ for } k \geq 4\text{)} \\
&= (k+1)! && \text{(by definition of factorial function)}
\end{aligned}
$$

- Therefore, by the principle of mathematical induction $P(n)$ is true for all integers n with $n \geq 4$.

Second Principle of Induction

"Strong Induction"

- Characterized by another inference rule:

$$\begin{array}{l} P(1) \\ \forall k \geq 1: (P(1) \wedge \overbrace{P(2) \wedge}^{P \text{ is true in } all \text{ previous cases}} \cdots \wedge P(k)) \rightarrow P(k+1) \\ \hline \therefore \forall n \geq 1: P(n) \end{array}$$

- The only difference between this and the 1st principle is that:
- the inductive step here makes use of the stronger hypothesis that all of $P(1)$, $P(2), \ldots, P(k)$ are true, not just $P(k)$.

Example of Second Principle

- Show that every integer $n > 1$ can be written as a product $n = p_1 p_2 \ldots p_s = \prod p_i$ of some series of s prime numbers.
- Let $P(n)$ = "n has that property "
- ***Basis step***: $n = 2$, let $s = 1$, $p_1 = 2$. Then $n = p_1$
- ***Inductive step***: Let $k \geq 2$. Assume $\forall 2 \leq i \leq k$: $P(i)$.
- Consider $k + 1$. If it's prime, let $s = 1$, $p_1 = k + 1$.
- Else $k + 1 = ab$, where $1 < a \leq k$ and $1 < b \leq k$.

Then $a = p_1 p_2 \ldots p_t$ and $b = q_1 q_2 \ldots q_u$.
(by Inductive Hypothesis) Then we have that $k + 1 =$

$p_1 p_2 \ldots p_t q_1 q_2 \ldots q_u$,

a product of $s = t + u$ primes.

Generalizing Strong Induction

- Handle cases where the inductive step is valid only for integers greater than a particular integer
- $P(n)$ is true for $\forall n \geq b$ (b: fixed integer)
- **BASIS STEP**: Verify that $P(b)$, $P(b+1), \ldots, P(b+j)$ are true (j: a fixed positive integer)
- **INDUCTIVE STEP**: Show that the conditional statement $[P(b) \wedge P(b+1) \wedge \cdots \wedge P(k)] \rightarrow P(k+1)$ is true for all positive integers $k \geq b + j$

2nd Principle example

- Prove that every amount of postage of 12 cents or more can be formed using just 4-cent and 5-cent stamps.
- $P(n)$ = "postage of n cents can be formed using 4-cent and 5-cent stamps" for $n \geq 12$.
- ***Basis step*:**
- $12 = 3 \cdot 4$
- $13 = 2 \cdot 4 + 1 \cdot 5$
- $14 = 1 \cdot 4 + 2 \cdot 5$
- $15 = 3 \cdot 5$
- So $\forall 12 \leq i \leq 15, P(i)$.

Example *(cont.)*

- ***Inductive step***:
- Let $k \geq 15$, assume $\forall 12 \leq i \leq k, P(i)$.
- Note $12 \leq k - 3 \leq k$, so $P(k - 3)$.

(by inductive hypothesis) This means we can form postage of k – 3 cents
using just 4-cent and 5-cent stamps.

- Add a 4-cent stamp to get postage for $k + 1$,

i.e. $P(k + 1)$ is true (postage of $k + 1$ cents
can be formed using 4-cent and 5-cent stamps).

- Therefore, by the 2nd principle of mathematical induction $P(n)$ is true for all integers n with $n \geq 12$.

Another 2nd Principle example

- Prove by the 1st Principle.
- $P(n)$ = "postage of n cents can be formed using 4-cent and 5-cent stamps", $n \geq 12$.
- ***Basis step*: $P(12)$: $12 = 3 \cdot 4$.**
- ***Inductive step*: $P(k) \rightarrow P(k+1)$**
- Case 1: At least one 4-cent stamp was used for $P(k)$
- $k + 1 = k - 4 + 5$ (i.e. replace the 4-cent stamp with a 5-cent stamp to form a postage of $k + 1$ cents)

Example *Continues*…

- ***Inductive step***: $P(k) \to P(k+1)$
- <u>Case 2</u>: No 4-cent stamps were used for $P(k)$
- Since $k \geq 12$, at least three 5-cent stamps are needed to form postage of k cents
- $k + 1 = k - 3 \cdot 5 + 4 \cdot 4$ (i.e. replace three 5-cent stamps with four 4-cent stamps to form a postage of $k + 1$ cents)

- Therefore, by the principle of mathematical induction $P(n)$ is true for all integers n with $n \geq 12$.

The Well-Ordering Property

- Another way to prove the validity of the inductive inference rule is by using the *well-ordering property*, which says that:
- Every non-empty set of non-negative integers has a minimum (smallest) element.
- $\forall \varnothing \subset S \subseteq \mathbf{N}$: $\exists m \in S$ such that $\forall n \in S$, $m \leq n$
- This implies that $\{n | \neg P(n)\}$ (if non-empty) has a minimum element m, but then the assumption that $P(m-1) \rightarrow P((m-1)+1)$ would be contradicted.

Lecture

Chapter 4. Induction and Recursion

4.3 Recursive Definitions and

Structural Induction

Recursive Definitions

- In induction, we *prove* all members of an infinite set satisfy some predicate *P* by:
  - proving the truth of the predicate for larger members in terms of that of smaller members.

- In ***recursive definitions***, we similarly *define* a function, a predicate, a set, or a more complex structure over an infinite domain (universe of discourse) by:
  - defining the function, predicate value, set membership, or structure of larger elements in terms of those of smaller ones.

Recursion

- ***Recursion*** is the general term for the practice of defining an object in terms of *itself*
  - or of part of itself.
  - This may seem circular, but it isn't necessarily.
- An inductive proof establishes the truth of $P(k+1)$ *recursively* in terms of $P(k)$.
- There are also recursive *algorithms*, *definitions*, *functions*, *sequences*, *sets*, and other structures.

Recursively Defined Functions

- Simplest case: One way to define a function $f:\mathbf{N}\rightarrow S$ (for any set S) or series $a_n = f(n)$ is to:
  - Define $f(0)$
  - For $n > 0$, define $f(n)$ in terms of $f(0),\ldots,f(n-1)$
- **Example**: Define the series $a_n = 2^n$ where n is a nonnegative integer recursively:
  - a_n looks like $2^0, 2^1, 2^2, 2^3,\ldots$
  - Let $a_0 = 1$
  - For $n > 0$, let $a_n = 2\cdot a_{n-1}$

Another Example

- Suppose we define $f(n)$ for all $n \in \mathbf{N}$ recursively by:
  - Let $f(0) = 3$
  - For all $n > 0$, let $f(n) = 2 \cdot f(n-1) + 3$
- What are the values of the following?
  - $f(1) =$ $2 \cdot f(0) + 3 = 2 \cdot 3 + 3 = 9$
  - $f(2) =$ $2 \cdot f(1) + 3 = 2 \cdot 9 + 3 = 21$
  - $f(3) =$ $2 \cdot f(2) + 3 = 2 \cdot 21 + 3 = 45$
  - $f(4) =$ $2 \cdot f(3) + 3 = 2 \cdot 45 + 3 = 93$

Recursive Definition of Factorial

- Give an inductive (recursive) definition of the factorial function,

  $$F(n) = n! = \prod_{1 \le i \le n} i = 1 \cdot 2 \cdots n$$

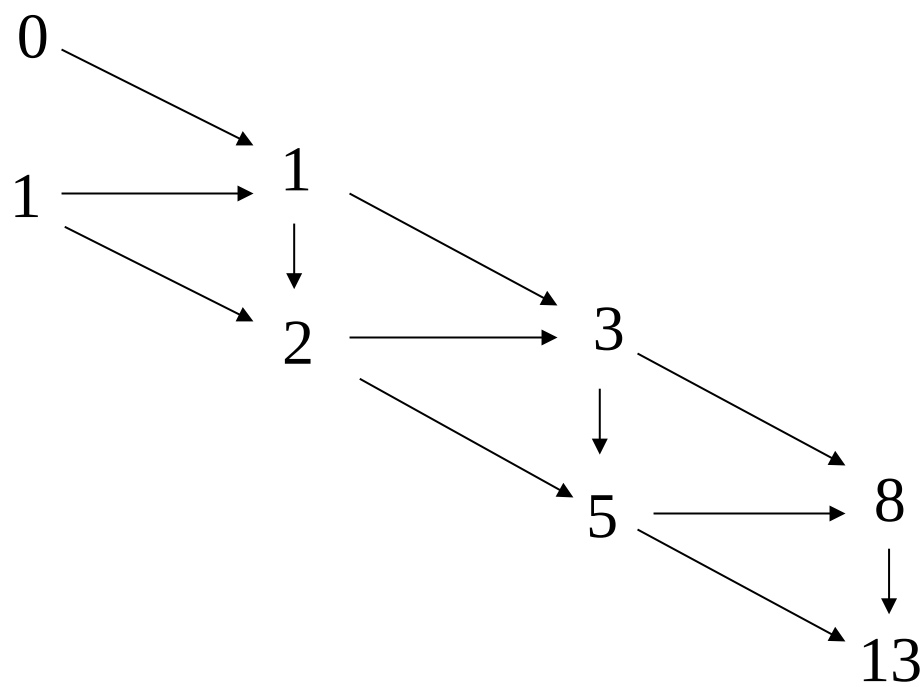
  - Basis step: $F(1) = 1$
  - Recursive step: $F(n) = n \cdot F(n-1)$ for $n > 1$
    - $F(2) = 2 \cdot F(1) = 2 \cdot 1 = 2$
    - $F(3) = 3 \cdot F(2) = 3 \cdot \{2 \cdot F(1)\} = 3 \cdot 2 \cdot 1 = 6$
    - $F(4) = 4 \cdot F(3) = 4 \cdot \{3 \cdot F(2)\} = 4 \cdot \{3 \cdot 2 \cdot F(1)\}$

      $= 4 \cdot 3 \cdot 2 \cdot 1 = 24$

The Fibonacci Numbers

- The ***Fibonacci numbers*** $f_{n\geq 0}$ is a famous series defined by:

$$f_0 = 0, \quad f_1 = 1, \quad f_{n\geq 2} = f_{n-1} + f_{n-2}$$



Inductive Proof about Fibonacci Numbers

- **Theorem:** $f_n < 2^n$. ← Implicitly for all $n \in \mathbf{N}$
- **Proof:** By induction
  - Basis step: $f_0 = 0 < 2^0 = 1$
    $f_1 = 1 < 2^1 = 2$
    Note: use of base cases of recursive definition
  - Inductive step: Use 2nd principle of induction (strong induction).

    Assume $\forall 0 \le i \le k,\ f_i < 2^i$. Then

    $$f_{k+1} = f_k + f_{k-1} \text{ is}$$
    $$< 2^k + 2^{k-1}$$
    $$< 2^k + 2^k = 2^{k+1}. \ \blacksquare$$

A Lower Bound on Fibonacci Numbers

- **Theorem:** For all integers $n \geq 3$, $f_n > \alpha^{n-2}$, where $\alpha = (1 + 5^{1/2})/2 \approx 1.61803$.
- **Proof.** (Using strong induction.)
  - Let $P(n) = (f_n > \alpha^{n-2})$.
  - **<u>Basis step</u>:**
    For $n = 3$, note that $\alpha^{n-2} = \alpha < 2 = f_3$.
    For $n = 4$, $\alpha^{n-2} = \alpha^2$

$$= (1 + 2 \cdot 5^{1/2} + 5)/4$$
$$= (3 + 5^{1/2})/2$$
$$\approx 2.61803 \qquad (= \alpha + 1)$$
$$< 3 = f_4.$$

A Lower Bound on Fibonacci

- **<u>Inductive step</u>:** For $k \geq 4$, assume $P(j)$ for $3 \leq j \leq k$, prove $P(k+1)$.
  - $f_{k+1} = f_k + f_{k-1} > \alpha^{k-2} + \alpha^{k-3}$ (by inductive hypothesis, $f_{k-1} > \alpha^{k-3}$ and $f_k > \alpha^{k-2}$).
  - Note that $\alpha^2 = \alpha + 1$.
    since $(3 + 5^{1/2})/2 = (1 + 5^{1/2})/2 + 1$
  - Thus, $\alpha^{k-1} = \alpha^2\alpha^{k-3} = (\alpha + 1)\alpha^{k-3}$
    $= \alpha\alpha^{k-3} + \alpha^{k-3} = \alpha^{k-2} + \alpha^{k-3}$.
  - So, $f_{k+1} = f_k + f_{k-1} > \alpha^{k-2} + \alpha^{k-3} = \alpha^{k-1}$.
  - Thus $P(k+1)$. ■

Recursively Defined Sets

- An infinite set *S* may be defined recursively, by giving:
  - A small finite set of *base* elements of *S*.
  - A rule for constructing new elements of *S* from previously-established elements.
  - Implicitly, *S* has no other elements but these.

- **Example:** Let $\overbrace{3\in S}^{\text{base element (basis step)}}$, and let $\overbrace{x+y\in S \text{ if } x,y\in S}^{\text{construction rule (recursive step)}}$.
  What is *S*?

Example *cont.*

- Let $3 \in S$, and let $x+y \in S$ if $x,y \in S$. What is S?
  - $3 \in S$ (*basis step*)
  - 6 (= 3 + 3) is in S (*first application of recursive step*)
  - 9 (= 3 + 6) and 12 (= 6 + 6) are in S (*second application of the recursive step*)
  - 15 (= 3 + 12 or 6 + 9), 18 (= 6 + 12 or 9 + 9), 21 (= 9 + 12), 24 (= 12 + 12) are in S (*third application of the recursive step*)
  - … so on
  - Therefore, **S = {3,** 6, 9, 12, 15, 18, 21, 24,…}
    = set of *all positive multiples of* 3

The Set of All Strings

- Given an alphabet **Σ**, the set **Σ*** of all strings over **Σ** can be recursively defined by:
  - **Basis step**: $\lambda \in \Sigma^*$ (λ : empty string)
  - **Recursive step**: $(w \in \Sigma^* \wedge x \in \Sigma) \rightarrow wx \in \Sigma^*$

- **Example**: If $\Sigma = \{0, 1\}$ then
  - $\lambda \in \Sigma^*$ (*basis step*)
  - 0 and 1 are in Σ^* (*first application of recursive step*)
  - 00, 01, 10, and 11 are in Σ^* (*second application of the recursive step*)
  - … so on
  - Therefore, Σ^* consists of all finite strings of 0's and 1's together with the empty string

String: Example

- Show that if $\Sigma = \{a, b\}$ then *aab* is in Σ^*.

  Proof: We construct it with a finite number of applications of the basis and recursive steps in the definition of Σ^*:

1. $\lambda \in \Sigma^*$ by the basis step.
2. By step 1, the recursive step in the definition of Σ^* and the fact that $a \in \Sigma$, we can conclude that $\lambda a = a \in \Sigma^*$.

Proof *cont.*

3. Since $a \in \Sigma^*$ from step 2, and $a \in \Sigma$, applying the recursive step again we conclude that $aa \in \Sigma^*$.

4. Since $aa \in \Sigma^*$ from step 3 and $b \in \Sigma$, applying the recursive step again we conclude that $aab \in \Sigma^*$.

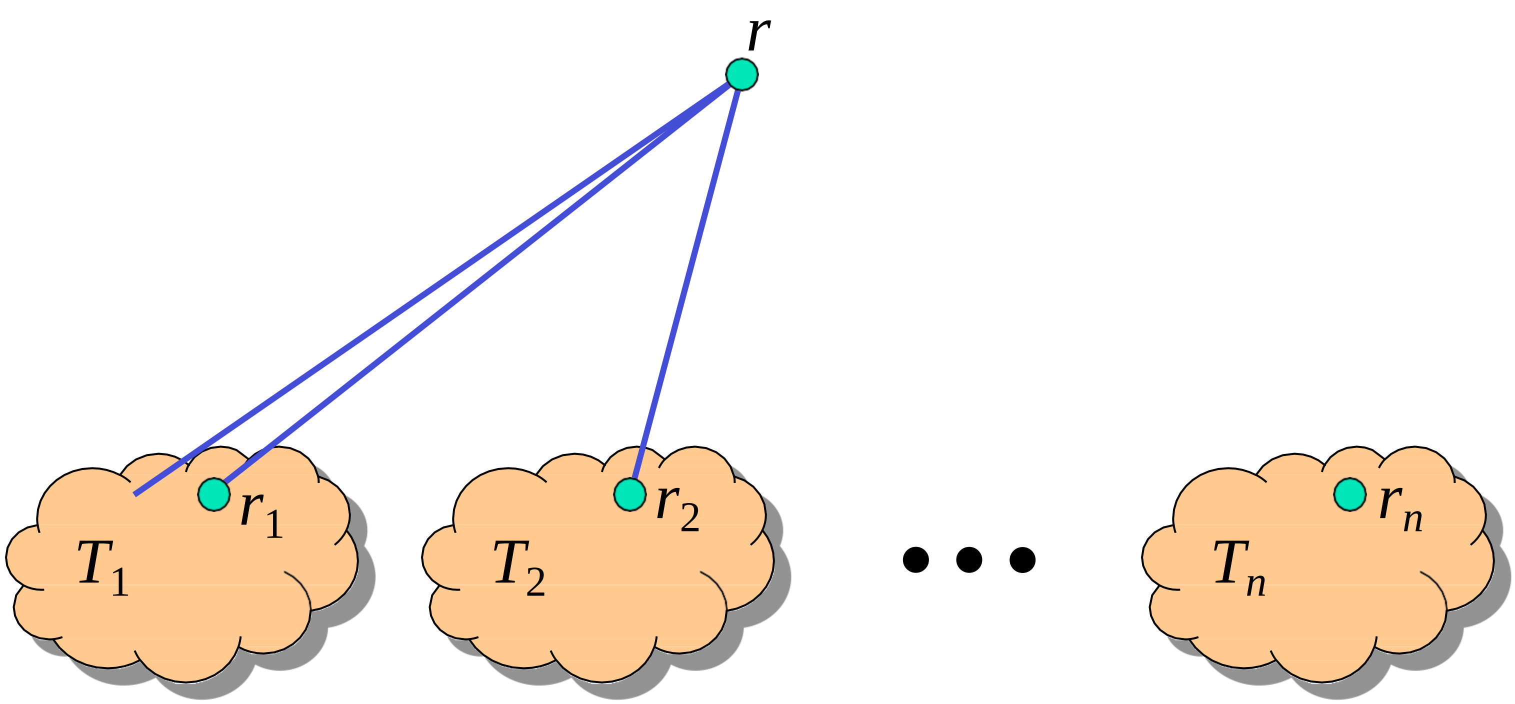
- Since we have shown $aab \in \Sigma^*$ with a finite number of applications of the basis and recursive steps in the definition we have finished the proof.

Rooted Trees

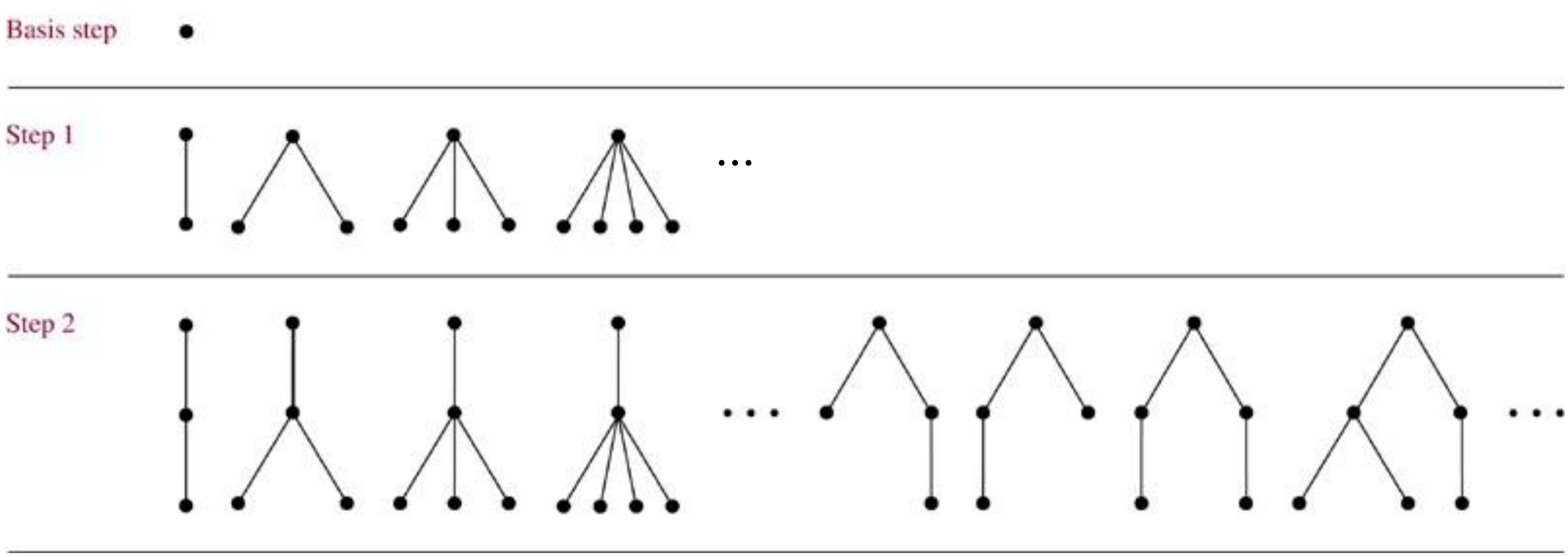
- Trees will be covered in more depth in chapter 10.
  - Briefly, a tree is a graph in which there is exactly one undirected path between each pair of nodes.
  - An undirected graph can be represented as a set of unordered pairs (called *arcs*) of objects called *nodes*.
- Definition of the set of rooted trees:
  - **Basis step**: Any single node r is a rooted tree.
  - **Recursive step**: If $T_1, \ldots, T_n$ are disjoint rooted trees with respective roots $r_1, \ldots, r_n$, and r is a node not in any of the T_i's, then another rooted tree is $\{(r, r_1), \ldots, (r, r_n)\} \cup T_1 \cup \cdots \cup T_n$.

Illustrating Rooted Tree

- How rooted trees can be combined to form a new rooted tree…



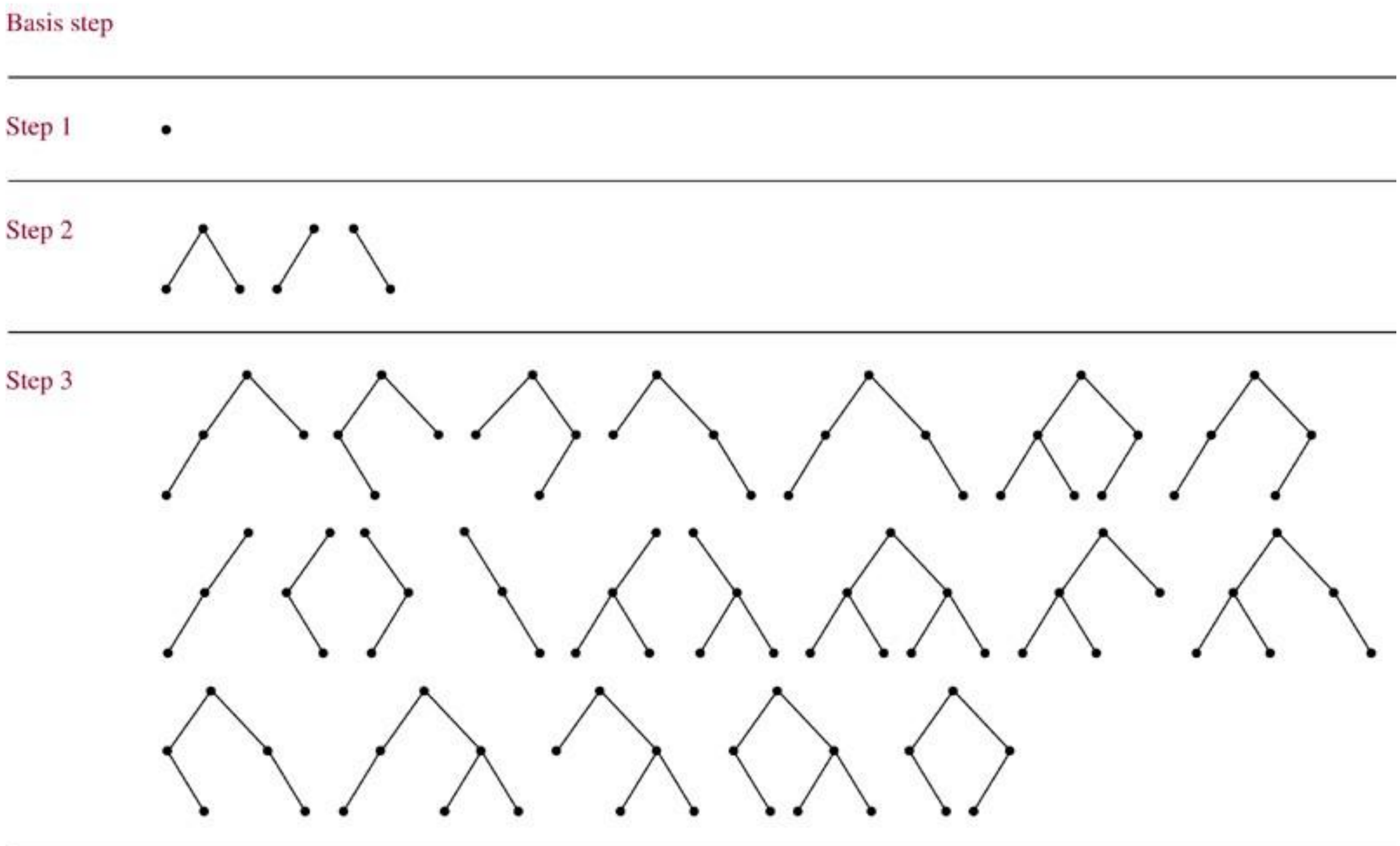
Building Up Rooted Trees



Extended Binary Trees

- A special case of rooted trees.
- Recursive definition of extended binary trees:
  - **Basis step**: The empty set $\emptyset$ is an extended binary tree.
  - **Recursive step**: If T_1, T_2 are disjoint extended binary trees, then $e_1 \cup e_2 \cup T_1 \cup T_2$ is an extended binary tree, where $e_1 = \emptyset$ if $T_1 = \emptyset$, and $e_1 = \{(r, r_1)\}$ if $T_1 \neq \emptyset$ and has root r_1, and similarly for e_2. (T_1 is the left subtree and T_2 is the right subtree.)

Building Up Extended BinaryTrees



Lamé's Theorem

- **Theorem:** $\forall a,b \in \mathbf{N}$, $a \geq b > 0$, and let n be the number of steps Euclid's algorithm needs to compute gcd(a,b).
  Then $n \leq 5k$, where $k = \lfloor \log_{10} b \rfloor + 1$ is the number of decimal digits in b.
  - Thus, Euclid's algorithm is linear-time in the number of digits in b.
    (or, Euclid's algorithm is O(log a))
- **Proof:**
  - Uses the Fibonacci sequence! (See next!)

Proof of Lamé's Theorem

- Consider the sequence of division-algorithm equations used in Euclid's alg.:

$$r_0 = r_1 q_1 + r_2 \quad \text{with } 0 \le r_2 < r_1$$
$$r_1 = r_2 q_2 + r_3 \quad \text{with } 0 \le r_3 < r_2$$
$$\ldots$$
$$r_{n-2} = r_{n-1} q_{n-1} + r_n \quad \text{with } 0 \le r_n < r_{n-1}$$
$$r_{n-1} = r_n q_n + r_{n+1} \quad \text{with } r_{n+1} = 0 \text{ (terminate)}$$

Where $a = r_0$, $b = r_1$, and $\gcd(a,b) = r_n$.

- The number of divisions (iterations) is n.

Continued on next slide…

Lamé Proof *cont.*

- Since $r_0 \geq r_1 > r_2 > \ldots > r_n$, each quotient $q_i \equiv \lfloor r_{i-1}/r_i \rfloor \geq 1$.
- Since $r_{n-1} = r_n q_n$ and $r_{n-1} > r_n$, $q_n \geq 2$.
- So we have the following relations between r and f:

$$r_n \geq 1 = f_2$$

$$r_{n-1} \geq 2r_n \geq 2f_2 = f_3$$

$$r_{n-2} \geq r_{n-1} + r_n \geq f_2 + f_3 = f_4$$

$$\ldots$$

$$r_2 \geq r_3 + r_4 \geq f_{n-1} + f_{n-2} = f_n$$

$$b = r_1 \geq r_2 + r_3 \geq f_n + f_{n-1} = f_{n+1}.$$

- Thus, if $n > 2$ divisions are used, then $b \geq f_{n+1} > \alpha^{n-1}$.
  - Thus, $\log_{10} b > \log_{10}(\alpha^{n-1}) = (n-1)\log_{10}\alpha \approx (n-1)0.208 > (n-1)/5$.
  - If b has k decimal digits, then $\log_{10} b < k$, so $n-1 < 5k$, so $n \leq 5k$.

Lecture

Chapter 4. Induction and Recursion

3. Recursive Definitions and Structural Induction
4. Recursive Algorithms

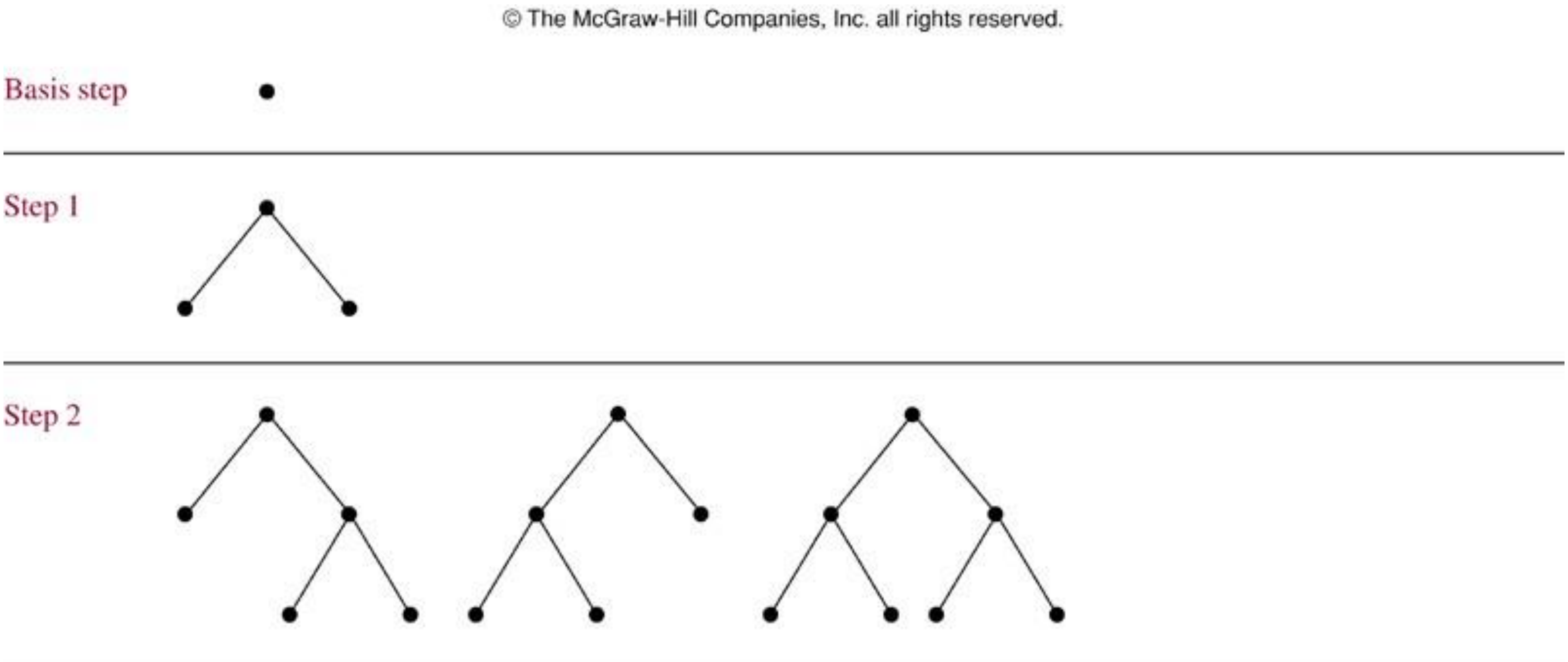
Review: Recursive Definitions

- ***Recursion*** is the general term for the practice of defining an object in terms of *itself* or of part of itself.
- In ***recursive definitions***, we similarly *define* a function, a predicate, a set, or a more complex structure over an infinite domain (universe of discourse) by:
  - defining the function, predicate value, set membership, or structure of larger elements in terms of those of smaller ones.

Full Binary Trees

- A special case of extended binary trees.
- Recursive definition of full binary trees:
  - **Basis step**: A single node r is a full binary tree.
    - Note this is different from the extended binary tree base case.
  - **Recursive step**: If T_1, T_2 are disjoint full binary trees with roots r_1 and r_2, then $\{(r, r_1), (r, r_2)\} \cup T_1 \cup T_2$ is an full binary tree.

Building Up Full Binary Trees



Structural Induction

- Proving something about a recursively defined object using an inductive proof whose structure mirrors the object's definition.
  - **Basis step**: Show that the result holds for all elements in the set specified in the basis step of the recursive definition
  - **Recursive step**: Show that if the statement is true for each of the elements in the new set constructed in the recursive step of the definition, the result holds for these new elements.

Structural Induction: Example

- Let $3 \in S$, and let $x+y \in S$ if $x,y \in S$.
  Show that S is the set of positive multiples of 3.
- Let $A = \{n \in \mathbf{Z}^+ | \ (3|n)\}$. We'll show that $A = S$.
  - **Proof:** We show that $A \subseteq S$ and $S \subseteq A$.
  - To show $A \subseteq S$, show $[n \in \mathbf{Z}^+ \wedge (3|n)] \rightarrow n \in S$.
    - **Inductive proof.** Let $n \in \mathbf{Z}^+$ and $P(n) = 3n \in S$.
      Induction over positive multiples of 3.

      Basis case: $n = 1$, thus $3 \in S$ by definition of S.

      Inductive step: Given $P(k)$, prove $P(k+1)$.
      By inductive hypothesis $3k \in S$, and $3 \in S$,
      so by definition of S, $3(k + 1) = 3k + 3 \in S$.

Example *cont.*

- To show $S \subseteq A$: let $n \in S$, show $n \in A$.
  - **Structural inductive proof.** Let $P(n) = n \in A$.

    Two cases: $n = 3$ (basis case), which is in A, or $n = x + y$ for $x,y \in S$ (recursive step).

    We know x and y are positive, since neither rule generates negative numbers.

    So, $x < n$ and $y < n$, and so we know x and y are in A, by strong inductive hypothesis.

    Since $3|x$ and $3|y$, we have $3|(x+y)$, thus $x + y = n \in A$. ■

Recursive Algorithms

- Recursive definitions can be used to describe functions and sets as well as *algorithms*.
- A *recursive procedure* is a procedure that invokes itself.
- A *recursive algorithm* is an algorithm that contains a recursive procedure.
- *An algorithm is called recursive if it solves a problem by reducing it to an instance of the same problem with smaller input.*

Example

- A procedure to compute a^n.

procedure *power*($a \neq 0$: real, $n \in \mathbf{N}$)

  if $n = 0$ **then return** 1

  else return $a \cdot power(a, n-1)$

problem → subproblems of the same type as the original problem

divide-and-conquer technique

Recursive Euclid's Algorithm

- gcd(a, b) = gcd((b mod a), a)

  procedure *gcd*(a,$b$$\in$**N** with $a < b$)

      if $a = 0$ **then return** b

      else return *gcd*(b **mod** a, a)

- Note recursive algorithms are often simpler to code than iterative ones…
- However, they can consume more stack space
  - if your compiler is not smart enough

Recursive Linear Search

{Finds x in series a at a location $\geq i$ and $\leq j$}

procedure *search*

(*a*: series; *i*, *j*: integer; *x*: item to find)

if $a_i = x$ **return** *i* {At the right item? Return it!}

if $i = j$ **return** 0 {No locations in range? Failure!}

return *search*(*a*, *i* +1, *j*, *x*) {Try rest of range}

- Note there is no real advantage to using recursion here over just looping

  for *loc* := *i* to *j*...

  recursion is slower because procedure call costs

Recursive Binary Search

{Find location of x in a, $\geq i$ and $\leq j$}

procedure *binarySearch*(a, x, i, j)

$m := \lfloor (i+j)/2 \rfloor$ {Go to halfway point}

if $x = a_m$ **return** m {Did we luck out?}

if $x < a_m \wedge i < m$ {If it's to the left, check that ½}

return *binarySearch*(a, x, i, $m-1$)

else if $x > a_m \wedge j > m$ {If it's to right, check that ½}

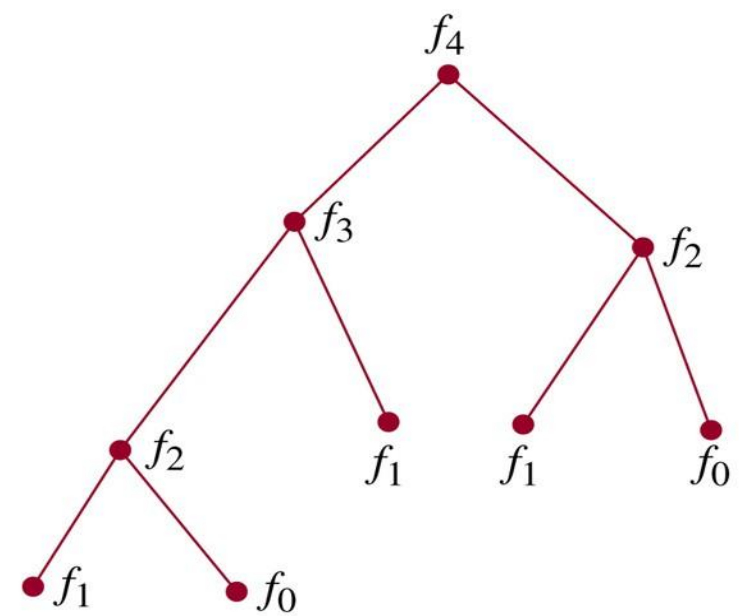
return *binarySearch*(a, x, $m+1$, j)

else return 0 {No more items, failure.}

Recursive Fibonacci Algorithm

procedure *fibonacci*($n \in \mathbf{N}$)
    if $n = 0$ **return** 0
    if $n = 1$ **return** 1
    return *fibonacci*($n - 1$) + *fibonacci*($n - 2$)

- Is this an efficient algorithm?
- How many additions are performed?



Analysis of Fibonacci Procedure

- **Theorem:** The recursive procedure *fibonacci*(n) performs $f_{n+1} - 1$ additions.
  - **Proof:** By strong structural induction over n, based on the procedure's own recursive definition.
    - **Basis step:**
      - *fibonacci*(0) performs 0 additions, and $f_{0+1} - 1 = f_1 - 1 = 1 - 1 = 0$.
      - Likewise, *fibonacci*(1) performs 0 additions, and $f_{1+1} - 1 = f_2 - 1 = 1 - 1 = 0$.

Analysis of Fibonacci Procedure

- **Inductive step**:

  $$fibonacci(k+1) = \underbrace{fibonacci(k)}_{\text{by } P(k):\ f_{k+1}-1 \text{ additions}} + \underbrace{fibonacci(k-1)}_{\text{by } P(k-1):\ f_k-1 \text{ additions}}$$

  - For $k > 1$, by strong inductive hypothesis, *fibonacci*(k) and *fibonacci*($k-1$) do $f_{k+1} - 1$ and $f_k - 1$ additions respectively.
  - *fibonacci*(k+1) adds 1 more, for a total of

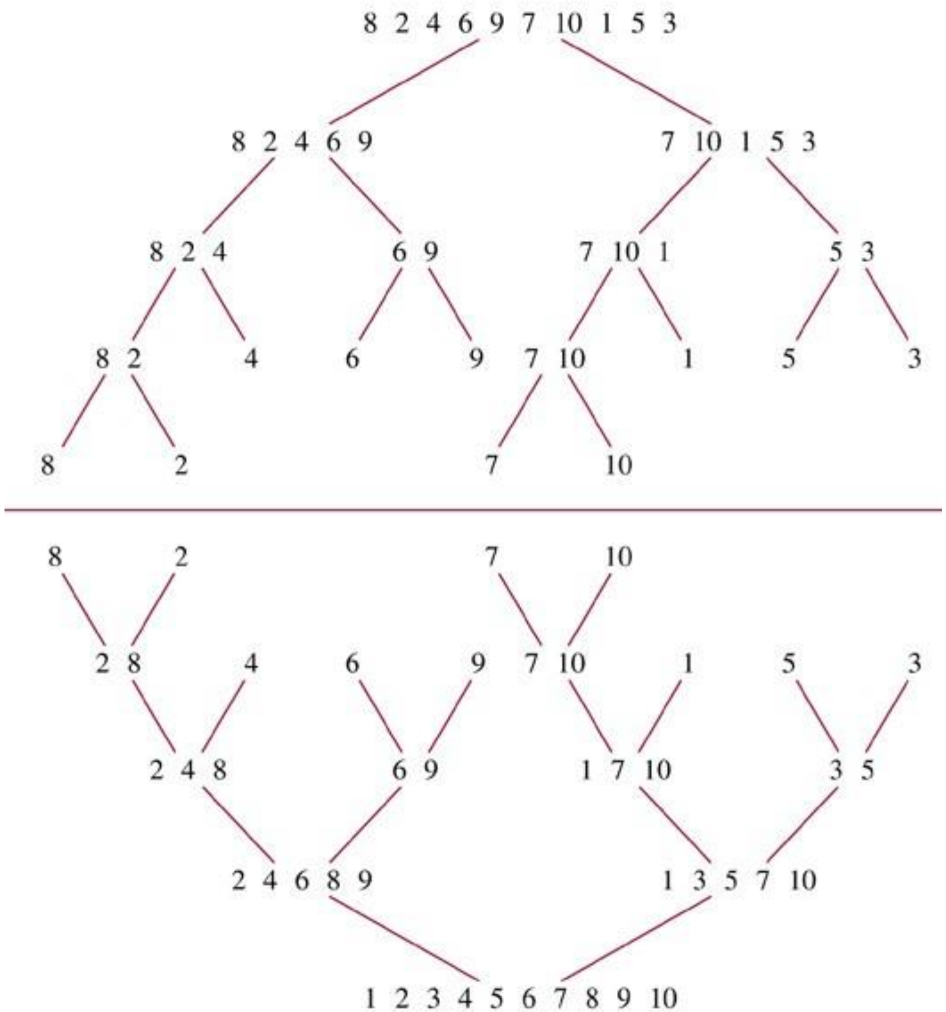
    $$(f_{k+1} - 1) + (f_k - 1) + 1 = f_{k+1} + f_k - 1$$
    $$= f_{k+2} - 1. \blacksquare$$

Iterative Fibonacci Algorithm

procedure *iterativeFib*($n \in \mathbf{N}$)
  if $n = 0$ **then**
    return 0
  else begin
    $x := 0$
    $y := 1$
    for $i := 1$ **to** $n - 1$ **begin**
      $z := x + y$
      $x := y$
      $y := z$
    end
  end
  return y {the nth Fibonacci number}

Requires only
$n - 1$ additions

Recursive Merge Sort Example



Split

Merge

Recursive Merge Sort

procedure *mergesort*($L = \ell_1, \ldots, \ell_n$)

  if $n > 1$ **then**

    $m := \lfloor n/2 \rfloor$ {this is rough ½-way point}

    $L_1 := \ell_1, \ldots, \ell_m$

    $L_2 := \ell_{m+1}, \ldots, \ell_n$

    $L := merge(mergesort(L_1), mergesort(L_2))$

  return L

- The merge takes $\Theta(n)$ steps, and therefore the merge-sort takes $\Theta(n \log n)$.

Merging Two Sorted Lists

TABLE 1 Merging the Two Sorted Lists 2, 3, 5, 6 and 1, 4.

First List	*Second List*	*Merged List*	*Comparison*
2 3 5 6	1 4		1 < 2
2 3 5 6	4	1	2 < 4
3 5 6	4	1 2	3 < 4
5 6	4	1 2 3	4 < 5
5 6		1 2 3 4	
		1 2 3 4 5 6	

Recursive Merge Method

{Given two sorted lists $A = (a_1,\ldots, a_{|A|})$, $B = (b_1,\ldots, b_{|B|})$, returns a sorted list of all.}

procedure *merge*(A, B: sorted lists)

if A = empty **return** B {If A is empty, it's B.}

if B = empty **return** A {If B is empty, it's A.}

if $a_1 < b_1$ **then**

return $(a_1, merge((a_2,\ldots, a_{|A|}), B))$

else

return $(b_1, merge(A, (b_2,\ldots, b_{|B|})))$

Efficiency of Recursive Algorithm

- The time complexity of a recursive algorithm may depend critically on the number of recursive calls it makes.

- **Example:** *Modular exponentiation* to a power n can take $\log(n)$ time if done right, but linear time if done slightly differently.
  - Task: Compute b^n **mod** m, where $m \geq 2$, $n \geq 0$, and $1 \leq b < m$.

Modular Exponentiation #1

- Uses the fact that $b^n = b \cdot b^{n-1}$ and that $x \cdot y \textbf{ mod } m = x \cdot (y \textbf{ mod } m) \textbf{ mod } m$.
  (Prove the latter theorem at home.)

  {Returns b^n **mod** m.}
  procedure *mpower*
  (b, n, m: integers with $m \geq 2$, $n \geq 0$, and $1 \leq b < m$)
  if $n=0$ **then return** 1 **else**
  return $(b \cdot mpower(b, n-1, m))$ **mod** m

- Note this algorithm takes $\Theta(n)$ steps!

Modular Exponentiation #2

- Uses the fact that $b^{2k} = b^{k\cdot 2} = (b^k)^2$.
- Then, b^{2k} **mod** $m = (b^k$ **mod** $m)^2$ **mod** m.

procedure *mpower*(b,n,m) {same signature}

if n=0 **then return** 1

else if 2|n **then**

  return *mpower*(b,n/2,m)^2 **mod** m

else return ($b\cdot$*mpower*(b,n−1,m)) **mod** m

- What is its time complexity? $\Theta(\log n)$ steps

A Slight Variation

- Nearly identical but takes $\Theta(n)$ time instead!

procedure $mpower(b,n,m)$ {same signature}

if $n=0$ **then return** 1

else if $2|n$ **then**

return $(mpower(b,n/2,m) \cdot$
$mpower(b,n/2,m))$ **mod** m

else return $(mpower(b,n-1,m) \cdot b)$ **mod** m

The number of recursive calls made is critical!

Lecture

Chapter 4. Induction and Recursion

4.5 Program Correctness

Chapter 5. Counting

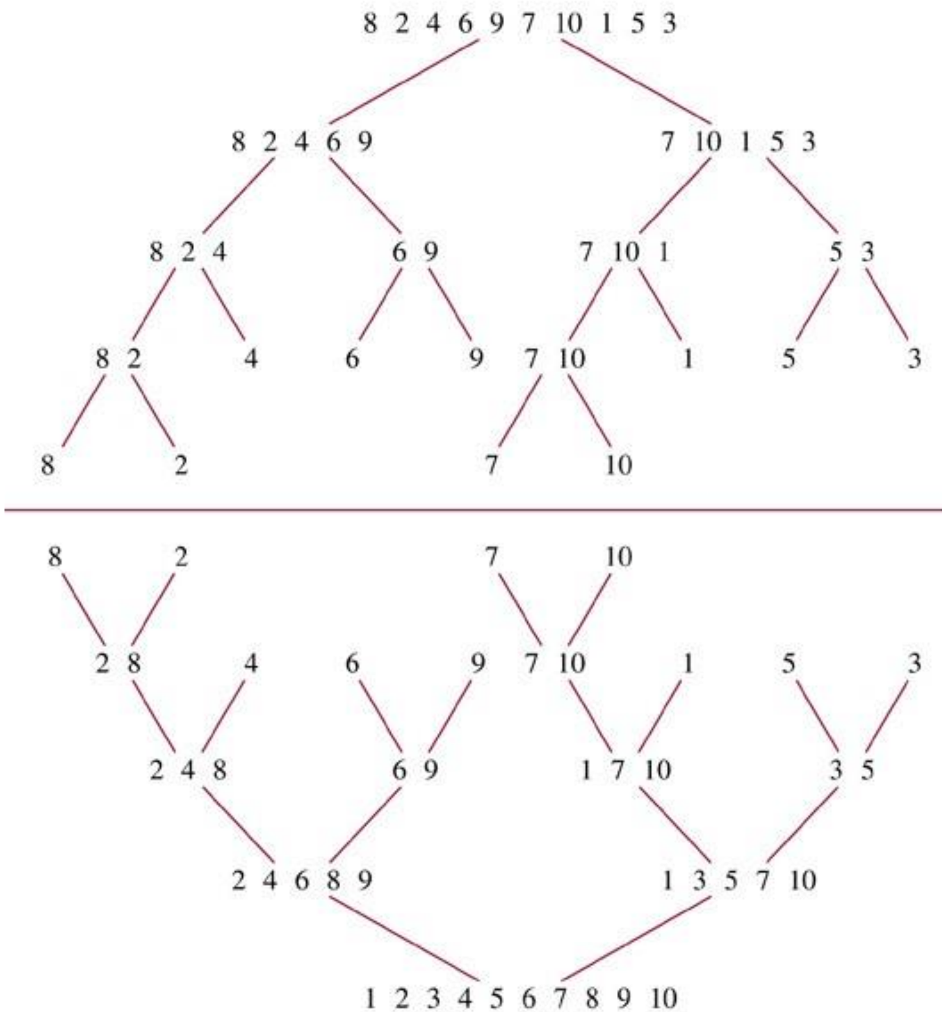
5.1 The Basics of Counting

Quiz

- Develop a recursive procedure for computing the minimum item in a list of integer numbers.

- Given is the recursive definition:
- $f(0) = f(1) = 2$
- $f(n+1) = f(n) * f(n-1)$
- Develop a recursive procedure for this definition
- What is your most time-efficient way to compute $f(n)$?
- What are the complexities of the recursive method and of yours?

Recursive Merge Sort



Split

Merge

Recursive Merge Sort

procedure *mergesort*($L = \ell_1, \ldots, \ell_n$)
if $n > 1$ **then**
$m := \leq\lfloor n/2 ƒ \rfloor$ {this is rough ½-way point}
$L_1 := \ell_1, \ldots, \ell_m$
$L_2 := \ell_{m+1}, \ldots, \ell_n$
$L := merge(mergesort(L_1), mergesort(L_2))$
return L

- The merge takes $\Theta(n)$ steps, and merge-sort takes $\Theta(n \log n)$.

Merging Two Sorted Lists

TABLE 1 Merging the Two Sorted Lists 2, 3, 5, 6 and 1, 4.

First List	*Second List*	*Merged List*	*Comparison*
2 3 5 6	1 4		1 < 2
2 3 5 6	4	1	2 < 4
3 5 6	4	1 2	3 < 4
5 6	4	1 2 3	4 < 5
5 6		1 2 3 4	
		1 2 3 4 5 6	

Recursive Merge Method

procedure *merge*(A, B: sorted lists)

{Given two sorted lists $A = (a_1, \ldots, a_{|A|})$, $B = (b_1, \ldots, b_{|B|})$, return a sorted list of all.}

if A = empty **return** B {If A is empty, it's B.}

if B = empty **return** A {If B is empty, it's A.}

if $a_1 < b_1$ **then**

$L := (a_1, merge((a_2, \ldots, a_{|A|}), B))$

else

$L := (b_1, merge(A, (b_2, \ldots, b_{|B|})))$

return L

Merge Routine

procedure *merge*(A, B: sorted lists)

L = empty list

i:=0, j:=0, k:=0

while $i < |A| \wedge j < |B|$ {$|A|$ is length of A}

if i=$|A|$ **then** $L_k := B_j$; $j := j + 1$

else if j=$|B|$ **then** $L_k := A_i$; $i := i + 1$

else if $A_i < B_j$ **then** $L_k := A_i$; $i := i + 1$

else $L_k := B_j$; $j := j + 1$

$k := k$+1

return L

Takes $\Theta(|A|+|B|)$ time

Program Correctness

- We want to be able to ***prove*** that a given program meets the intended specifications.
- This can often be done manually, or even by *automated program verification* tools.

- A program is ***correct*** if it produces the correct output for every possible input.
- A program is ***partially correct*** if it produces the correct output for every input for which the program eventually halts.

Initial & Final Assertions

- A program's I/O specification can be given using ***initial*** and ***final*** assertions.
  - **The *initial assertion p*** is the condition that the program's input (its initial state) is guaranteed to satisfy (by its user).
  - **The *final assertion q*** is the condition that the output produced by the program (in its final state) is required to satisfy.
- ***Hoare triple*** notation:
  - The notation ***p*{*S*}*q*** means that, for all inputs *I* such that *p*(*I*) is true, if program *S* (given input *I*) halts and produces output *O* = *S*(*I*), then *q*(*O*) is true.
- That is, *S* is partially correct with respect to specification *p*, *q*.

A Trivial Example

- Let S be the program fragment "$y := 2; z := x + y$"
- Let p be the initial assertion "$x = 1$".
  - The variable x will hold 1 in all initial states.
- Let q be the final assertion "$z = 3$".
  - The variable z must hold 3 in all final states.
- Prove $p\{S\}q$.
  - **Proof:** If $x = 1$ in the program's input state, then after running $y := 2$ and $z := x + y$, z will be $1 + 2 = 3$.

Hoare Triple Inference Rules

- Deduction rules for Hoare Triple statements.
- A simple example: the ***composition rule***: $p\{S_1\}q$

$q\{S_2\}r$

$\therefore\ p\{S_1;\ S_2\}r$

- **It says:** If program S_1 given condition p produces condition q, and S_2 given q produces r, then the program "S_1 followed by S_2", if given p, yields r.

Inference Rule for *if* Statements

- Program segment that is the conditional statement

if *condition* **then**
    S

- Rule of inference

$$\frac{\begin{array}{l}(p \wedge condition)\{S\}q \\ (p \wedge \neg condition) \to q\end{array}}{\therefore p\{\textbf{if } condition \textbf{ then } S\}q}$$

Initial assertion

Final assertion

- Example: Show that **T** {**if** $x > y$ **then** $y := x$} $y \geq x$.

Proof: When the initial assertion is true and if $x > y$, then the **if** body is executed, which sets $y = x$,
and so afterwards $y \geq x$ is true.
Otherwise, $x \leq y$ and so $y \geq x$. In either case $y \geq x$ is true. So the fragment meets the specification.

if-then-else Rule

- Program segment that is the conditional statement **if** *condition* **then**

S_1
else
S_2

- Rule of inference

$$\frac{\begin{array}{l}(p \wedge condition)\{S_1\}q \\ (p \wedge \neg condition)\{S_2\}q\end{array}}{\therefore\ p\{\textbf{if}\ condition\ \textbf{then}\ S_1\ \textbf{else}\ S_2\}q}$$

- Example: Show that

$\mathbf{T}$ {**if** $x < 0$ **then** $abs := -x$ **else** $abs := x$} $abs = |x|$

- If $x < 0$ then after the **if** body, $abs = -x = |x|$.

If $\neg(x < 0)$, *i.e.*, $x \geq 0$, then after the **else** body, $abs = x = |x|$. So the rule applies and the program segment is correct.

Loop Invariants

- For a while loop "**while** *condition* S", we say that p is a ***loop invariant*** of this loop if $(p \wedge condition)\{S\}p$.
- If p (and the continuation condition *condition*) is true before executing the body, then p remains true afterwards.
- And so p stays true through *all* subsequent iterations.

- This leads to the inference rule:

p is a loop invariant

$$\frac{(p \wedge condition)\{S\}p}{\therefore\ p\{\textbf{while}\ condition\ S\}(\neg condition \wedge p)}$$

Loop Invariant Example

S

$$
\left.\begin{array}{l}
i := 1 \\
fact := 1 \\
\textbf{while } i < n \\
\qquad i := i + 1; \\
\qquad fact := fact \cdot i \\
\textbf{end while}
\end{array}\right.
$$

- Prove that the following Hoare triple holds when n is a positive integer: **T** $\{S\}$ $(fact = n!)$
  - **Proof.** Note that p: "$fact = i! \land i \le n$" is a loop invariant, and is true before the loop. Thus, after the loop we have $(\neg condition \land p) \Leftrightarrow \neg(i < n) \land (fact = i! \land i \le n) \Leftrightarrow i = n \land fact = i! \Leftrightarrow fact = n!$. ■

Big Example

- S **=** S_1**;** S_2**;** S_3**;** S_4 (compute the product of two integers m, n)

procedure *multiply*(m, n: integers) — p: $m, n \in \mathbf{Z}$

S_1 **if** $n < 0$ **then** $a := -n$ **else** $a := n$ — q: $p \wedge (a = |n|)$

S_2 $k := 0; x := 0$ — r: $q \wedge (k = 0) \wedge (x = 0)$

Loop invariant $x = mk \wedge k \leq a$

S_3
while $k < a$ {

$x = x + m; k = k + 1$

}

Maintains loop invariant:

$x = mk \wedge k \leq a \quad x = mk \wedge k = a$

$\therefore x = ma = m|n|$ — s

$\therefore (n < 0 \wedge x = -mn) \vee (n \geq 0 \wedge x = mn)$

S_4 **if** $n < 0$ **then** *prod* $:= -x$ **else** *prod* $:= x$ — $prod = mn$ — t

Chapter 5: Counting

- Combinatorics
- The study of the number of ways to put things together into various combinations.

- *E.g.* In a contest entered by 100 people,
- how many different top-10 outcomes could occur?
- *E.g.* If a password is 6~8 letters and/or digits,
- how many passwords can there be?

Sum and Product Rules

- Let m be the number of ways to do task 1 and n the number of ways to do task 2,
- with each number independent of how the other task is done,
- and also assume that no way to do task 1 simultaneously also accomplishes task 2.
- Then, we have the following rules:
- The ***sum rule***: The task "do either task 1 or task 2, but not both" can be done in $m + n$ ways.
- The ***product rule***: The task "do both task 1 and task 2" can be done in mn ways.

The Sum Rule

- If a task can be done in one of n_1 ways, or in one of n_2 ways, …, or in one of n_m ways, where none of the set of n_i ways of doing the task is the same as any of the set of n_j ways, for all pairs i and j with $1 \leq i < j \leq m$.
- Then the number of ways to do the task is

$n_1 + \ \ n_2 + \cdots + n_m$.

The Sum Rule: Example 1

- A student can choose a computer project from one of three lists A, B, and C:
- List A: 23 possible projects
- List B: 15 possible projects
- List C: 19 possible projects
- No project is on more than one list
- How many possible projects are there to choose from?

$$23 + 15 + 19 = 57$$

The Sum Rule: Example 2

- What is the value of k after the following code has been executed?

$k := 0$

for $i_1 := 1$ **to** n_1 $k := k + 1$

for $i_2 := 1$ **to** n_2

$k := k + 1$

…

for $i_m := 1$ **to** n_m

$k := k + 1$

$$n_1 + n_2 + \cdots + n_m$$

The Product Rule

- Suppose that a procedure can be broken down into a sequence of m successive tasks.

If the task T_1 can be done in n_1 ways;
the task T_2 can then be done in n_2 ways; ...; and
the task T_m can be done in n_m ways, then there are $n_1 \cdot n_2 \cdots n_m$ ways to do the procedure.

The Product Rule: Example

- Show that a set $\{x_1, \ldots, x_n\}$ containing n elements has 2^n subsets.
- A subset can be constructed in

n successive steps:

- Pick or do not pick x_1, pick or do not pick x_2, …, pick or do not pick x_n .
  - Each step can be done in two ways.
- Thus the number of possible subsets is $\underbrace{2 \cdot 2 \cdots 2}_{n \text{ factors}} = 2^n$.

Lecture

Chapter 5. Counting

1. The Basics of Counting
2. The Pigeonhole Principle
3. Permutations and Combinations

Review

- **<u>Sum Rule</u>:** If a task can be done in one of n_1 ways, or in one of n_2 ways, …, or in one of n_m ways, where none of the set of n_i ways of doing the task is the same as any of the set of n_j ways, for all pairs i and j with $1 \le i < j \le m$. Then the number of ways to do the task is $n_1 + n_2 + \cdots + n_m$.
- **<u>Product Rule</u>:** Suppose that a procedure can be broken down into a sequence of m successive tasks. If the task T_1 can be done in n_1 ways; the task T_2 can then be done in n_2 ways; ...; and the task T_m can be done in n_m ways, then there are $n_1 \cdot n_2 \cdots n_m$ ways to do the procedure.

The Product Rule: Example

- Show that a set $\{x_1,\ldots, x_n\}$ containing n elements has 2^n subsets.
  - A subset can be constructed in n successive steps:
    - Pick or do not pick x_1, pick or do not pick x_2, …, pick or do not pick x_n .
  - Each step can be done in two ways.
  - Thus the number of possible subsets is $\underbrace{2\cdot 2\cdots 2}_{n \text{ factors}} = 2^n$.

The Product Rule: Example

- What is the value of k after the following code has been executed?

$k := 0$

for $i_1 := 1$ **to** n_1

for $i_2 := 1$ **to** n_2

…

for $i_m := 1$ **to** n_m

$k := k + 1$

$n_1 \cdot n_2 \cdots n_m$

The Product Rule: Example

- How many functions are there from a set with *m* elements to one with *n* elements?

$$n^m$$

- How many one-to-one functions are there from a set with *m* elements to one with *n* elements?

$$n \cdot (n - 1)(n - 2) \cdots (n - m + 1)$$

- More examples in the textbook

IP Address Example

- In version 4 of the Internet Protocol (IPv4)
  - Internet address is a string of 32 bits
  - Network number (*netid*) + host number (*hostid*)
  - Valid computer addresses are in one of 3 types:
    - A ***class A*** IP address consists of 0, followed by a 7-bit "netid" $\neq \mathbf{1}^7$, and a 24-bit "hostid"
    - A ***class B*** address has 10, followed by a 14-bit netid and a 16-bit hostid.
    - A ***class C*** address has 110, followed by a 21-bit netid and an 8-bit hostid.
  - The 3 classes have distinct headers (0, 10, 110)
  - Hostids that are all 0s or all 1s are not allowed.

128.171.224.100

IP Address Example

Bit Number	0	1	2	3	4	8	16	24	31
Class A	0	netid				hostid			
Class B	1	0	netid				hostid		
Class C	1	1	0	netid				hostid	
Class D	1	1	1	0	Multicast Address				
Class E	1	1	1	1	0	Address			

- How many valid computer addresses are there?

IP Address Solution

- \# of addresses

  = (# class A) + (# class B) + (# class C)

  (by sum rule)

- \# class A = (# valid netids)$\times$(# valid hostids)

  (by product rule)

- \# valid class A netids = $2^7 - 1 = 127$.

- \# valid class A hostids = $2^{24} - 2 = 16{,}777{,}214$.

- Continuing in this fashion we find the answer is:

  3,737,091,842 (3.7 billion IP addresses)

Set Theoretic Version

- If A is the set of ways to do task 1, and B the set of ways to do task 2, and if A and B are disjoint, then:
  - The ways to do either task 1 or 2 are $A \cup B$, and $|A \cup B| = |A| + |B|$
  - The ways to do both task 1 and 2 can be represented as $A \times B$, and $|A \times B| = |A| \cdot |B|$

Inclusion-Exclusion Principle

- Suppose that k out of m ways of doing task 1 also simultaneously accomplish task 2.
  - And there are also n ways of doing task 2.
- Then, the number of ways to accomplish "Do either task 1 or task 2" is $m + n - k$.
- Set theory: If A and B are not disjoint, then $|A \cup B| = |A| + |B| - |A \cap B|$.
  - If they are disjoint, this simplifies to $|A| + |B|$.

Inclusion-Exclusion Example

- Some hypothetical rules for passwords:
  - Passwords must be 2 characters long
  - Each character must be a letter a ~ z, a digit 0 ~ 9, or one of the 10 punctuation characters ! @ # $ % ^ & * ( )
  - Each password must contain <u>*at least one*</u> digit or punctuation character

Setup of Problem

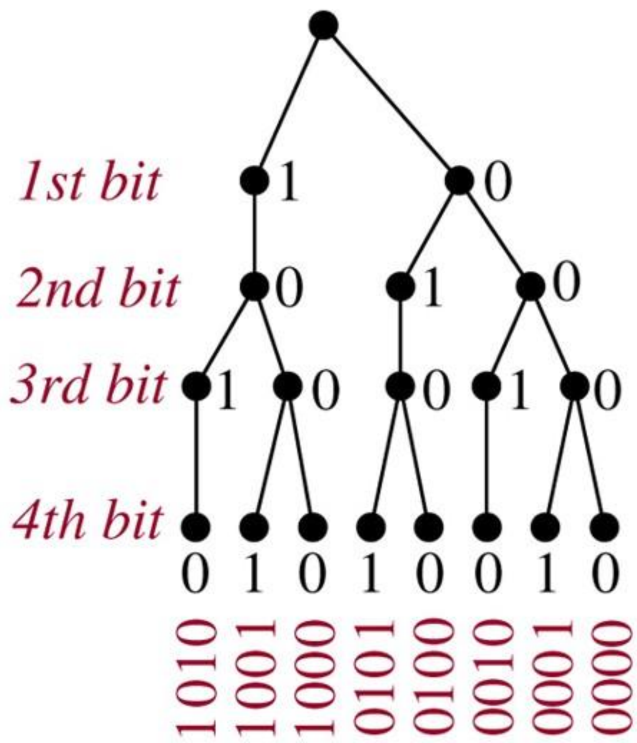
- A legal password has a digit or punctuation character in position 1 **or** position 2.
  - These cases overlap, so the principle applies.
- # of passwords with OK symbol in position #1 $= (10 + 10) \times (10 + 10 + 26) = 20 \times 46 = 920$
- # with OK symbol in pos. #2 $= 46 \times 20 = 920$
- # with OK symbol both places $= 20 \times 20 = 400$
- Answer: $920 + 920 - 400 = 1{,}440$

Tree Diagrams

- A tree diagram can be used in many different counting problems.
- To use trees in counting, we use a branch to represent each possible choice.
- We represent the possible outcomes by the leaves, which are the endpoints of branches not having other branches starting at them.

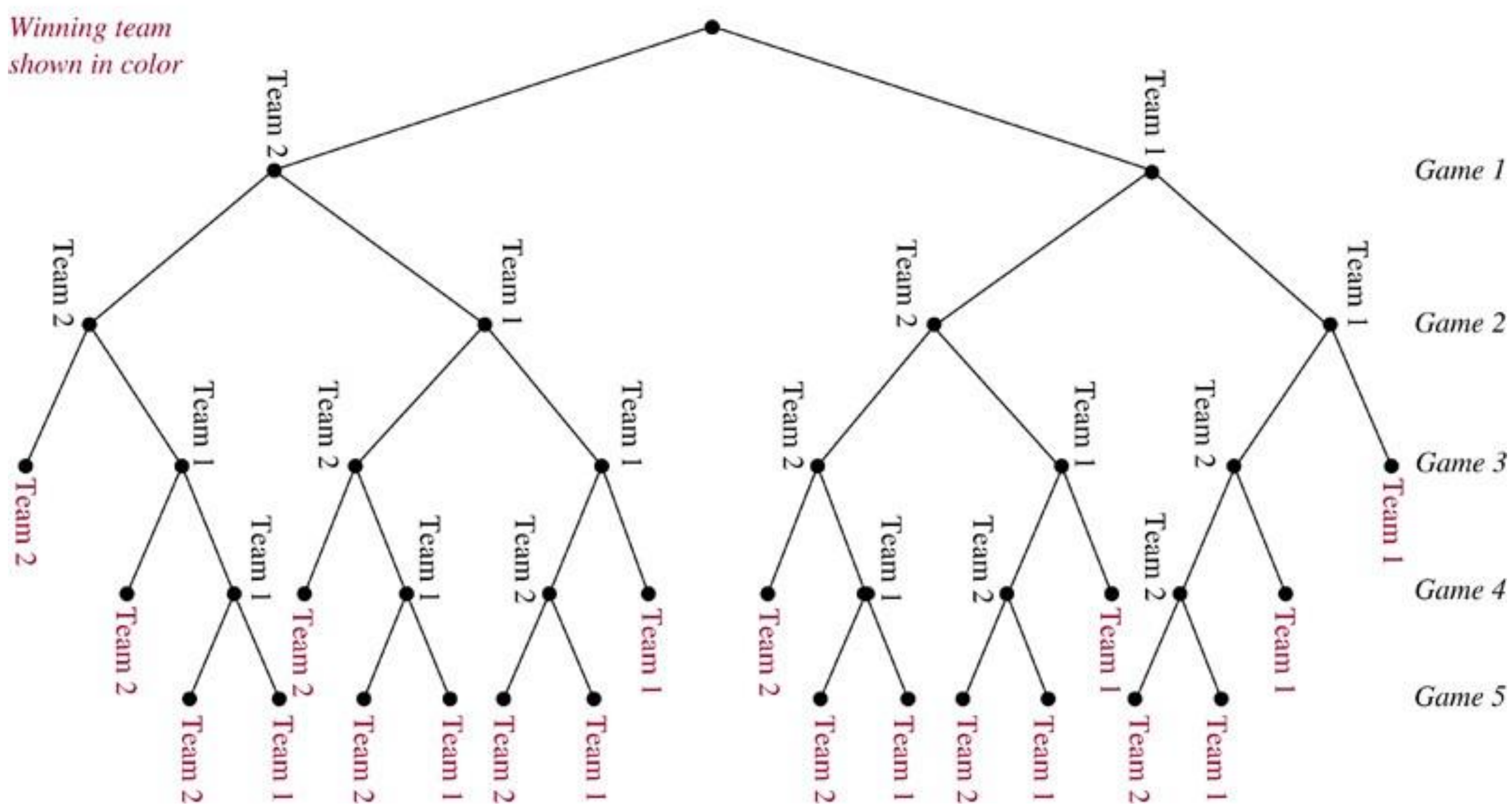
Tree Diagrams: Example

How many bit strings of length four do not have two consecutive 1s?



Tree Diagrams: Example

- A playoff between two teams consists of at most five games. The first team that wins three games wins the playoff. In how many different ways can the playoff occur?

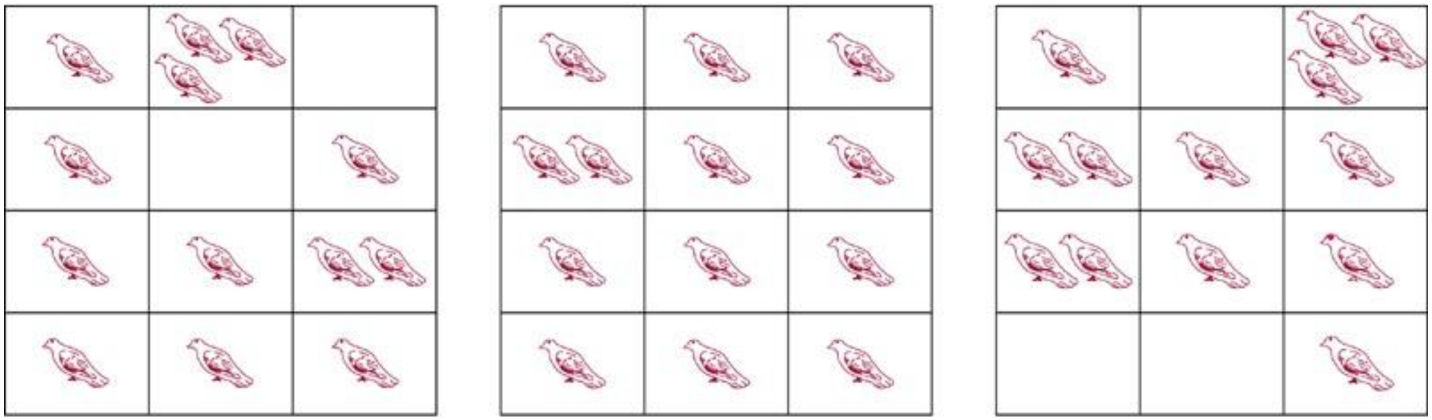


The Pigeonhole Principle

- A.k.a. the "Dirichlet drawer principle" or the "Shoe Box Principle".
- If $k + 1$ or more objects are assigned to k places, then at least 1 place must be assigned 2 or more objects.
- In terms of the assignment function:
  - If $f: A \to B$ and $|A| \geq |B| + 1$, then some element of B has more than two preimages under f.
    - I.e., f is not one-to-one.

The Pigeonhole Principle

- Proof by contradiction:
  - If the conclusion is false, each pigeonhole contains at most one pigeon and in this time, we can account for at most *k* pigeons.
  - Since there are *k* + 1 pigeons, we have a contradiction.



(a) (b) (c)

Pigeonhole Principle: Example

- There are 101 possible numeric grades (0% ~ 100%) rounded to the nearest integer.
  - Also, there are >101 students in a class.
- Therefore, there must be at least one (rounded) grade that will be shared by at least 2 students at the end of the semester.
  - i.e., the function from students to rounded grades is *not* a one-to-one function.

Another Example of P.P.

- 10 persons have first names as Alice, Bernare, and Charles, and last names as Lee, McDuff, and Ng. Show that at least two persons have the same first and last names.
- Solution:
  - 9 possible names for the 10 persons → 10 pigeons and 9 pigeonholes.
  - Assignment of names to people = assignment of pigeonholes to the pigeons
  - By the Pigeonhole Principle, some name (pigeonhole) is assigned to at least two persons (pigeons).

Generalized Pigeonhole Principle

- If N objects are assigned to k places, then at least one place must be assigned at least $\lceil N/k \rceil$ objects.
- *E.g.*, there are $N = 280$ students in a class. There are $k = 52$ weeks in the year.
  - Therefore, there must be at least 1 week during which at least $\lceil 280/52 \rceil = \lceil 5.38 \rceil = 6$ students in the class have a birthday.

Proof of G.P.P.

- By contradiction.

  Suppose every place has $< \lceil N/k \rceil$ objects,

  thus $\leq \lceil N/k \rceil - 1$.

- Then the total number of objects is at most

$$k\left(\left\lceil \frac{N}{k} \right\rceil - 1\right) < k\left(\left(\frac{N}{k} + 1\right) - 1\right) = k\left(\frac{N}{k}\right) = N$$

- So, there are less than *N* objects, which contradicts our assumption of *N* objects! ■

G.P.P. Example I

- Given: There are 280 students in a class.
  - Without knowing anybody's birthday, what is the largest value of *n* for which we can prove using the G.P.P. that at least *n* students must have been born in the same month?
- Answer:

$$\lceil 280/12 \rceil = \lceil 23.3 \rceil = 24$$

G.P.P. Example II

- What is the least number of area codes needed to guarantee that the 25 million phones in a state can be assigned distinct 10-digit telephone numbers?
  - Phone #: NXX-NXX-XXXX
    - N: 2 ~ 9 and X: any digit
- Solution
  - NXX-XXXX: $(8 \cdot 10 \cdot 10) \cdot (10 \cdot 10 \cdot 10 \cdot 10) = 8$ million
  - By G.P.P. at least ϒ$\lceil 25{,}000{,}000/8{,}000{,}000 \rceil = 4$ phones have the identical numbers
  - Hence, at least 4 area codes are required

Fun Pigeonhole Proof

- **Example 4:** $\forall n \in \mathbf{N}$, $\exists$ a multiple $m > 0$ of n such that m has only 0's and 1's in its decimal expansion!

- **Proof:** Consider the n+1 decimal integers 1, 11, 111, …, $\underbrace{1\cdots1}_{n+1}$. They have only n possible remainders mod n.

  So, take the difference of two that have the same remainder. The result is the answer! □

A Specific Case

- Let n=3. Consider 1, 11, 111, 1111.
  - 1 mod 3 = 1
  - 11 mod 3 = 2
  - 111 mod 3 = 0 ⇐ Lucky extra solution.
  - 1,111 mod 3 = 1

Note same residue

- 1,111 − 1 = 1,110 = 3·370.
  - It has only 0's and 1's in its expansion.
  - Its remainder mod 3 = 0, so it's a multiple of 3.

Baseball Example

- Suppose that next June, the Marlins baseball team plays at least 1 game a day, but ≤ 45 games total. Show there must be some sequence of consecutive days in June during which they play *exactly* 14 games.
  - **Proof:** Let a_j be the number of games played on or before day j. Then, $a_1,\ldots,a_{30} \in \mathbf{Z}^+$ is a sequence of 30 distinct integers with $1 \leq a_j \leq 45$.
    Therefore $a_1+14,\ldots,a_{30}+14$ is a sequence of 30 distinct integers with $15 \leq a_j+14 \leq 59$.
    Thus, $(a_1,\ldots,a_{30},a_1+14,\ldots,a_{30}+14)$ is a sequence of 60 integers from the set {1,..,59}.
    By the Pigeonhole Principle, two of them must be equal, but $a_i \neq a_j$ for $i \neq j$. So, $\exists ij$: $a_i = a_j+14$.
    Thus, 14 games were played on days $a_j+1, \ldots, a_i$.

Baseball Problem Illustrated

- Example of $\{a_i\}$: Note all elements are distinct.
  - 1, 2, 4, 5, 7, 8, 10, 11, 13, 14, 16, 17, 19, 21, 22, 23, 25, 27, 29, 30, 31, 33, 34, 36, 37, 39, 40, 41, 43, 45
  - Then $\{a_i+14\}$ is the following sequence: 15, 16, 18, 19, 21, 22, 24, 25, 27, 28, 30, 32, 33, 35, 36, 37, 39, 41, 43, 44, 45, 47, 48, 50, 51, 53, 54, 55, 57, 59

Thus, for example, exactly 14 games were played during days
3 to 11:
2+1+2+1+2+1+2+1+2

- In any 60 integers from 1-59 there must be some duplicates, indeed we find the following ones:
  - 16, 19, 21, 22, 25, 27, 30, 33, 36, 37, 39, 41, 43, 45

Lecture

Chapter 5. Counting

3. Permutations and Combinations
4. Binomial Coefficients
5. Generalized Permutations and Combinations

Permutations

- A ***permutation*** of a set S of distinct elements is an **ordered** sequence that contains each element in S exactly once.
- E.g. $\{A, B, C\} \rightarrow$ six permutations:

ABC, *ACB*, *BAC*, *BCA*, *CAB*, *CBA*

- An **ordered** arrangement of r distinct elements of S is called an ***r-permutation of S***.
- The number of r-permutations of a set with $n = |S|$ elements is

$$\boxed{P(n,r) = n(n-1)(n-2)\cdots(n-r+1) = \frac{n!}{(n-r)!}, \quad 0 \le r \le n.}$$

- $P(n,n) = n!/(n-n)! = n!/0! = n!$ (Note: $0! = 1$)

Permutation Examples

- **Example**: Let $S = \{1, 2, 3\}$.
  - The arrangement 3, 1, 2 is a permutation of S ($3! = 6$ ways)
  - The arrangement 3, 2 is a 2-permutation of S ($3\cdot 2=3!/1! = 6$ ways)

- **Example**: There is an armed nuclear bomb planted in your city, and it is your job to disable it by cutting wires to the trigger device. There are 10 wires to the device.

  If you cut exactly the right three wires, in exactly the right order, you will disable the bomb, otherwise it will explode!

  If the wires all look the same, what are your chances of survival?

$P(10,3) = 10\times 9\times 8 = 720$,
so there is a 1 in 720 chance that you'll survive!

More Permutation Examples

- **Example 6**: Suppose that a sales woman has to visit eight different cities. She must begin her trip in a specified city, but she can visit the other seven cities in any order she wishes. How many possible orders can the saleswoman use when visiting these cities?

> First city is determined, and the remaining seven can be ordered arbitrarily: $7! = 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 5040$

- **Example 7**: How many permutations of the letters ABCDEFGH contain the string ABC?

> ABC must occur as a block, i.e. consider it as one object
> Then, it'll be the number of permutations of six objects (ABC, D, E, F, G, H), which is $6! = 720$

Another Example

- How many ways are there to pick a set of 3 people from a group of 6?
- There are 6 choices for the first person, 5 for the second one, and 4 for the third one, so there are $6 \cdot 5 \cdot 4 = 120$ ways to do this.
- This is not the correct result!
- For example, picking person C, then person A, and then person E leads to the same group as first picking E, then C, and then A.
- However, these cases are counted separately in the above equation.
- So how can we compute how many different subsets of people can be picked (that is, we want to disregard the order of picking)?

Combinations

- **How many different committees of three students can be formed from a group of four students?**
- An ***r-combination*** of elements of a set S is an **unordered** selection of r elements from the set. Thus, an r-combination is simply a subset $T \subseteq S$ with r members, $|T| = r$.
- **Example**: S = {1, 2, 3, 4}, then {1, 3, 4} is a 3-combination from S
- **Example**: How many distinct 7-card hands can be drawn from a standard 52-card deck?
- The order of cards in a hand doesn't matter.

Calculate *C*(*n*, *r*)

- Consider that we can obtain the *r*-permutation of a set in the following way:
- First, we form all the *r*-combinations of the set (there are $C(n, r)$ such *r*-combinations)
- Then, we generate all possible orderings in each of these *r*-combinations (there are $P(r, r)$ such orderings in each case).
- Therefore, we have:

$$P(n, r) = C(n, r) \cdot P(r, r)$$

$$C(n, r) = \frac{P(n, r)}{P(r, r)} = \frac{n(n-1)\cdots(n-r+1)}{r!}$$

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

Combinations

- The number of *r*-combinations of a set with $n = |S|$ elements is

$$C(n, r) = \binom{n}{r} = \frac{P(n, r)}{P(r, r)} = \frac{n!/(n-r)!}{r!} = \frac{n!}{r!(n-r)!}$$

- Note that $C(n, r) = C(n, n - r)$
  - Because choosing the *r* members of *T* is the same thing as choosing the $(n - r)$ non-members of *T*.

$$C(n, n-r) = \binom{n}{n-r} = \frac{n!}{(n-r)![n-(n-r)]!} = \frac{n!}{(n-r)!r!}$$

Combination Example I

- How many distinct 7-card hands can be drawn from a standard 52-card deck?
- The order of cards in a hand doesn't matter.

- Answer:

$C(52, 7) = P(52, 7) / P(7, 7) = 52! / (7!\cdot 45!)$

$= (52\cdot 51\cdot 50\cdot 49\cdot 48\cdot 47\cdot 46) / (7\cdot 6\cdot 5\cdot 4\cdot 3\cdot 2\cdot 1)$

17 10 7 $\cancel{8}$

$\cancel{2}$

$52\cdot 17\cdot 10\cdot 7\cdot 47\cdot 46 = 133{,}784{,}560$

Combination Example II

- $C(4, 3) = 4$, since, for example, the 3-combinations of a set {1, 2, 3, 4} are {1, 2, 3}, {1, 3, 4}, {2, 3, 4}, {1, 2, 4}.
- $C(4, 3) = P(4, 3) / P(3, 3) = 4! / (3! \times 1!)$

$= (4 \times 3 \times 2) / (3 \times 2 \times 1) = 4$

- How many ways are there to pick a set of 3 people from a group of 6 (disregarding the order of picking)?
- $C(6, 3) = 6! / (3! \times 3!)$

$= (6 \times 5 \times 4) / (3 \times 2 \times 1) = 20$

- There are 20 different groups to be picked

Combination Example III

- A soccer club has 8 female and 7 male members. For today's match, the coach wants to have 6 female and 5 male players on the grass. How many possible configurations are there?

- $C(8, 6) \times C(7, 5)$

$= \{P(8, 6) / P(6, 6)\} \times \{P(7, 5) / P(5, 5)\}$

$= \{8!/ (2! \times 6!)\} \times \{7!/ (2! \times 5!)\}$

$= \{(8 \times 7) / 2!\} \times \{(7 \times 6) / 2!\}$

$= 28 \times 21$

$= 588$

Binomial Coefficients

- Expressions of the form $C(n, r)$ are also called ***binomial coefficients***
- Coefficients of the expansion of powers of binomial expressions
- Binomial expression is a simply the sum of two terms such as $x + y$
- Example:

$$(x+y)^3 = (x+y)(x+y)(x+y)$$

$$= (xx + xy + yx + yy)(x+y)$$

$$= xxx + xxy + xyx + xyy + yxx + yxy + yyx + yyy$$

$$= C(3,0)x^3 + C(3,1)x^2 y + C(3,2)xy^2 + C(3,3)\,y^3$$

$$= x^3 + 3x^2 y + 3xy^2 + y^3$$

The Binomial Theorem

THE BINOMIAL THEOREM Let x and y be variables, and let n be a nonnegative integer. Then

$$(x+y)^n = \sum_{j=0}^{n} \binom{n}{j} x^{n-j} y^j = \binom{n}{0} x^n + \binom{n}{1} x^{n-1} y + \cdots + \binom{n}{n-1} x y^{n-1} + \binom{n}{n} y^n.$$

Proof: We use a combinatorial proof. The terms in the product when it is expanded are of the form $x^{n-j}y^j$ for $j = 0, 1, 2, \ldots, n$. To count the number of terms of the form $x^{n-j}y^j$, note that to obtain such a term it is necessary to choose $n - j$ xs from the n sums (so that the other j terms in the product are ys). Therefore, the coefficient of $x^{n-j}y^j$ is $\binom{n}{n-j}$, which is equal to $\binom{n}{j}$. This proves the theorem. ◁

Examples

- $(a + b)^9 \rightarrow$ the coefficient of $a^5b^4 = C(9, 4)$
- The coefficient of $x^{12}y^{13}$ in the expansion of $(2x - 3y)^{25}$
- By binomial theorem

$$\left(2x + (-3y)\right)^{25} = \sum_{j=0}^{25} \binom{25}{j} (2x)^{25-j} (-3y)^j$$

  - The coefficient of $x^{12}y^{13}$ is obtained when $j = 13$

$$C(25,13) \cdot 2^{12} \cdot (-3)^{13} = -\frac{25!}{13! \cdot 12!} 2^{12} 3^{13}$$

- $(x + y + z)^9 \rightarrow$ the coefficient of $x^2y^3z^4 = C(9, 2) \cdot C(7, 3).C(9,4)$.

COROLLARY 1

COROLLARY 1 Let n be a nonnegative integer. Then

$$\sum_{k=0}^{n} \binom{n}{k} = 2^n.$$

Proof: Using the binomial theorem with $x = 1$ and $y = 1$, we see that

$$2^n = (1+1)^n = \sum_{k=0}^{n} \binom{n}{k} 1^k 1^{n-k} = \sum_{k=0}^{n} \binom{n}{k}.$$

This is the desired result. ◁

There is also a nice combinatorial proof of Corollary 1, which we now present.

Proof: A set with n elements has a total of 2^n different subsets. Each subset has zero elements, one element, two elements, . . . , or n elements in it. There are $\binom{n}{0}$ subsets with zero elements, $\binom{n}{1}$ subsets with one element, $\binom{n}{2}$ subsets with two elements, . . . , and $\binom{n}{n}$ subsets with n elements. Therefore,

$$\sum_{k=0}^{n} \binom{n}{k}$$

counts the total number of subsets of a set with n elements. By equating the two formulas we have for the number of subsets of a set with n elements, we see that

$$\sum_{k=0}^{n} \binom{n}{k} = 2^n.$$

◁

COROLLARY 2

COROLLARY 2 Let n be a positive integer. Then

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

Proof: When we use the binomial theorem with $x = -1$ and $y = 1$, we see that

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

This proves the corollary.

Remark: Corollary 2 implies that

$$\binom{n}{0}+\binom{n}{2}+\binom{n}{4}+\cdots = \binom{n}{1}+\binom{n}{3}+\binom{n}{5}+\cdots.$$

COROLLARY 3

Let n be a nonnegative integer. Then

$$\sum_{k=0}^{n} 2^k \binom{n}{k} = 3^n.$$

Proof: We recognize that the left-hand side of this formula is the expansion of $(1+2)^n$ provided by the binomial theorem. Therefore, by the binomial theorem, we see that

$$(1+2)^n = \sum_{k=0}^{n} \binom{n}{k} 1^{n-k} 2^k = \sum_{k=0}^{n} \binom{n}{k} 2^k.$$

Hence

$$\sum_{k=0}^{n} 2^k \binom{n}{k} = 3^n.$$

◁

Generalized Permutations Combinations

- Permutations and combinations allowing repetitions.
  - How many strings of length *r* can be formed from the English alphabet?
  - How many different ways are possible when we select a dozen donuts from a box that contains four different kinds of donuts?
- Permutations where not all objects are distinguishable.
  - The number of ways we can rearrange the letters of the word *MISSISSIPPI*

Permutations with Repetitions

- **<u>Theorem 1</u>**: The number of *r*-permutations of n objects with repetition allowed is n^r.
- Proof: There are *n* ways to select an element of the set for each of *r* positions with repetition allowed. By the product rule, the answer is given as *r* multiples of *n*.

- **Example**: How many strings of length *r* can be formed from the English alphabet?
- Answer: 26^r

Combinations with Repetitions

- An example
- How many ways are there to select four pieces of fruit from a bowl containing apples, oranges, and pears if there are at least four pieces of each type of fruit in the bowl?
- In this case, the order in which the pieces are selected does not matter, only the types of fruit, not the individual piece, matter.

Combinations with Repetitions

- Example Rephrased: The number of 4-combinations with repetition allowed from a 3-element set {apple, orange, pear}
  - All four in same type: 4 apples, 4 oranges, 4 pears [3 ways]
  - Three in same type: two cases for each of 3 apples, 3 oranges, 3 pears [2*3=6 ways]
  - Two diff. pairs with each pair in same type [3 ways]
  - Only one pair in same type [3 ways]
  - <u>Total 15 ways</u>
- Can be generalized:
  - The number of ways to fill 4 slots from 3 categories with repetition allowed

Combinations with Repetitions

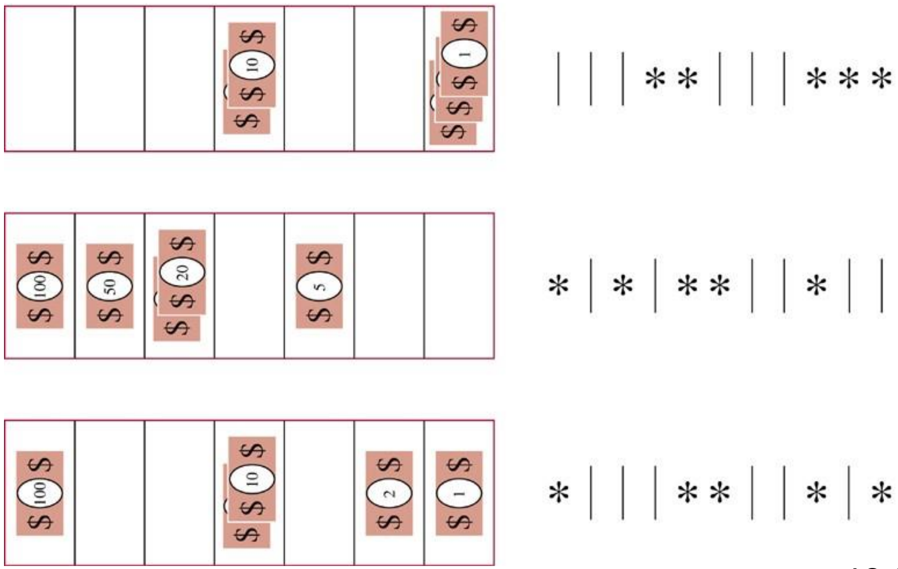
4 apples
3 apples, 1 orange
3 oranges, 1 pear
2 apples, 2 oranges
2 apples, 1 orange, 1 pear

4 oranges
3 apples, 1 pear
3pears, 1 apple
2 apples, 2 pears
2 oranges, 1apple, 1 pear

4 pears
3 oranges, 1 apple
3 pears, 1 orange
2 pears, 2 oranges
1apple, 1 orange, 2 pears

Example

- How many ways are there to select five bills from a cash box containing $1 bills, $2 bills, $5 bills, $10 bills, $20 bills, $50 bills, and $100 bills?
- The order in which the bills are chosen doesn't matter
- The bills of each denomination are indistinguishable
- At least five bills of each type



- $C(7–1+5, 5)$

$= C(11, 5)$

$= 11! / (5!\cdot 6!)$

$= 462$

Combinations with Repetitions

- **<u>Theorem 2</u>**: The number of *r*-combinations from a set with *n* elements with repetition allowed is:

$$\boxed{C(n+r-1,r)=C(n+r-1,n-1)}$$

- Other representations with the same meaning
- # of ways to fill *r* slots from *n* categories with repetition allowed
- # of ways to select *r* elements from *n* categories of elements with repetition allowed

Proof of Theorem 2

- Represent each r-combinations from a set with n elements with repetition allowed by <u>a list of $n-1$ bars and r stars</u>.
- $n-1$ bars: used to mark off n different cells (categories)
- r stars: each star in *i-th* cell (if any) represents an element that is selected for the *i-th* category

- # of different lists that containing $n-1$ bars and r stars
- = # of ways to chose the r positions to place the r stars from $n+r-1$ positions [$\mathbf{C(n+r-1, r)}$]
- = # of ways to chose the $n-1$ positions to place the $n-1$ bars from $n+r-1$ positions [$\mathbf{C(n+r-1, n-1)}$]

More Examples

- How many ways can I fill a box holding 100 pieces of candy from 30 different types of candy?
  - Solution: Here #stars = 100, #bars = 30 – 1, so there are $C(100+29,100) = 129! / (100! \cdot 29!)$ different ways to fill the box.

- How many ways if I must have at least 1 piece of each type?
  - Solution: Now, we are reducing the #stars to choose over to (100 – 30) stars, so there are $C(70+29, 70) = 99! / (70!29!)$

When to Use Generalized Combinations

- Besides categorizing a problem based on its order and repetition requirements as a generalized combination, there are a couple of other characteristics which help us sort:
  - In generalized combinations, having all the slots filled in by only selections from one category is allowed;
  - It is possible to have more slots than categories.

More Integer Solutions & Restrictions

- How many integer solutions are there to:

  $a + b + c + d = 15$,

  when $a \geq -3$, $b \geq 0$, $c \geq -2$ and $d \geq -1$?

- In this case, we alter the restrictions and equation so that the restrictions "go away." To do this, we need each restriction ≥ 0 and balance the number of slots accordingly.
- Hence $a \geq -3+3$, $b \geq 0$, $c \geq -2+2$ and $d \geq -1+1$, yields $a + b + c + d = 15+3+2+1 = 21$
- So, there are C(21+4–1,21) = C(24,21) = C(24,3) = (24x23x22)/(3x2) = 2024 solutions.

Distributing Objects into Distinguishable Boxes

- Distinguishable (or labeled) objects to distinguishable boxes
  - How many ways are there to distribute hands of <u>5 cards</u> to each of <u>four players</u> from the standard deck of 52 cards?

C(52,5)C(47,5)C(42,5)C(37,5)

- Indistinguishable (or unlabeled) objects to distinguishable boxes
  - How many ways are there to place <u>10 indistinguishable balls</u> into <u>8 distinguishable bins</u>?

C(8+10-1, 10) = C(17,10) = 17! / (10!7!)

Distributing Distinguishable Objects into Indistinguishable Boxes

- How many ways are there to put 4 different employees into 3 indistinguishable offices, when each office can contain any number of employees?
- all four in one office: C(4,4) = 1
- three + one: C(4,3) = 4
- two + two: C(4,2)/2 = 3
- two + one + one: C(4,2) = 6

Distributing Indistinguishable Objects into Indistinguishable Boxes

- How many ways are there to pack 6 copies of same book into 4 identical boxes, where each box can contain as many as six books?
- List # of books in each box with the largest # of books, followed by #s of books in each box containing at least 1 book, in order of decreasing # of books in a box.

6

5,1

4,2 4,1,1

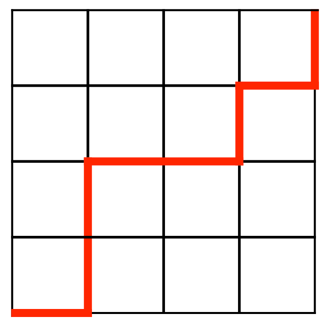
3,3 3,2,1 3,1,1,1

2,2,2 2,2,1,1

Another Combination Example

- How many routes are there from the lower left corner of an $n \times n$ square grid to the upper right corner if we are restricted to traveling only to the right or upward.
- Solution

R: right
U: up



- route → *RUURRURU* : a string of *n* *R*'s and *n* *U*'s
- Any such string can be obtained by selecting *n* positions for the *R*'s, without regard to the order of selection, from among the 2*n* available positions in the string and then filling the remaining position with *U*'s.
- Thus there are C(2*n*,*n*) possible routes.