Introduction to Set Theory

Textbook Reading: First three sections of Chapter 1 up to page 20.

Introduction to Set Theory

- A **set** is a structure representing an **unordered** collection (group, plurality) of zero or more **distinct** (different) objects.
- **Set theory** deals with operations between, relations among, and statements about sets.

Basic notations for sets

- For sets, we'll use variables *S*, *T*, *U*, *A*, *B*, …
- We can denote a set *S* in writing by listing all of its elements in curly braces:
  - {a, b, c} is the set of whatever 3 objects are denoted by a, b, c.
- *Set builder notation*: For any proposition *P*(*x*) over any universe of discourse, {*x* : *P*(*x*)} is *the set of all x such that P(x). Also denoted* {*x* | *P*(*x*)}

  e.g., {*x* : *x* is an integer where *x*>0 and *x*<5 }

Basic properties of sets

- Sets are inherently ***unordered***:
  - No matter what objects a, b, and c denote,
    {a, b, c} = {a, c, b} = {b, a, c} =
    {b, c, a} = {c, a, b} = {c, b, a}.
- All elements are ***distinct*** (unequal); multiple listings make no difference!
  - {a, b, c} = {a, a, b, a, b, c, c, c, c}.
  - This set contains at most 3 elements!

Definition of Set Equality

- Two sets are declared to be equal *if and only if* they contain <u>exactly the same</u> elements.
- In particular, it does not matter *how the set is defined or denoted.*
- For example: The set {1, 2, 3, 4} =

  {*x* | *x* is an integer where *x* > 0 and *x* < 5 } =

  {*x* | *x* is a positive integer whose square

  is > 0 and < 25}

Basic Set Relations: Member of

- $x \in S$ ("*x* is in *S*") is the proposition that object *x* is an $\in$*lement* or *member* of set *S*.
- $x \notin S$ ("*x* is not in *S*") is the proposition that object *x* is **not** an $\in$*lement* of set *S*.
- For $\in$xample
  - $3 \in \{1,2,3,4,5\}$,
  - 'a' $\in \{x \mid x$ is a letter of the alphabet$\}$
  - $2 \notin \{1,3,5,7,9,11,13\}$
  - 'a' $\notin \{x \mid x$ is a capital letter of the alphabet$\}$

Logical symbols

$\forall$	for all
$\rightarrow$	implies
$\leftrightarrow$	if and only if (iff)
$\exists$	there exists
$\nexists$	there does not exist
$\wedge$	and
$\vee$	or

Set equality

Two sets are equal **iff** (if and only if) they have all the same members.”

Can define set equality in terms of $\in$ relation:

$$\forall S,T\colon S = T \leftrightarrow (\forall x\colon x \in S \leftrightarrow x \in T)$$

The Empty Set

- $\varnothing$ ("null", "the empty set") is the unique set that contains no elements whatsoever.
- $\varnothing = \{\}$

Subset and Superset Relations

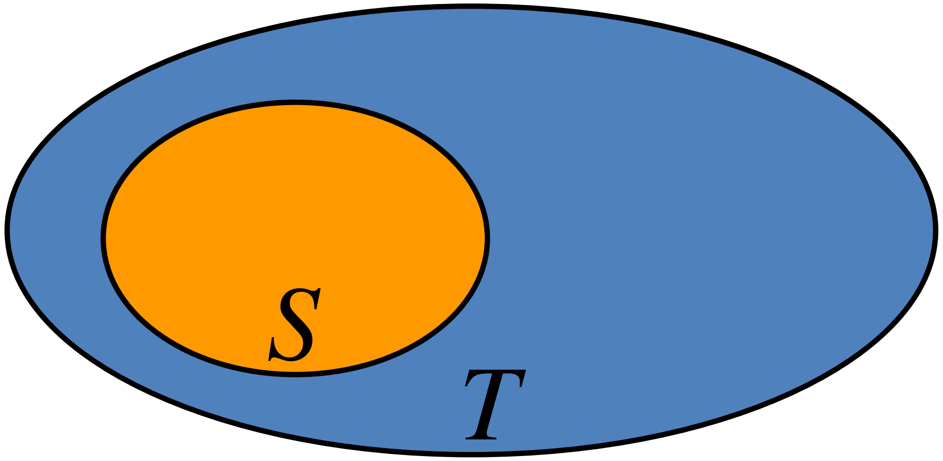
- $S \subseteq T$ ("S is a *subset* of T") means that every element of S is also an element of T.
- Equivalently,

$$S \subseteq T \leftrightarrow \forall x\, (x \in S \rightarrow x \in T)$$

- $\varnothing \subseteq S$, $S \subseteq S$.
- $S \supseteq T$ ("S is a *superset* of T") means $T \subseteq S$.
- Note $S = T \leftrightarrow S \subseteq T \wedge S \supseteq T$.
- $S \not\subseteq T$ iff $\exists x$ such that $x \in S \wedge x \notin T$

Proper (Strict) Subsets & Supersets

- $S \subset T$ ("S is a *proper subset* of T") means that $S \subseteq T$ but $S \neq T$. Similar for $S \supset T$.



Example:
$\{1,2\} \subset \{1,2,3\}$

Venn Diagram equivalent of $S \subset T$

Sets Are Objects, Too!

- The objects that are elements of a set may *themselves* be sets.
- For example, let $S=\{x \mid x \subseteq \{1,2,3\}\}$ then

$S=\{\emptyset, \{1\}, \{2\}, \{3\},\{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}$

- Note that $1 \neq \{1\} \neq \{\{1\}\}$!!!!

Cardinality

- $|S|$ (read "the *cardinality of S*") is a measure of how many different elements *S* has. *e.g.,*

$$|\varnothing|=0, \quad |\{1,2,3\}| = 3, \quad |\{a,b\}| = 2,$$

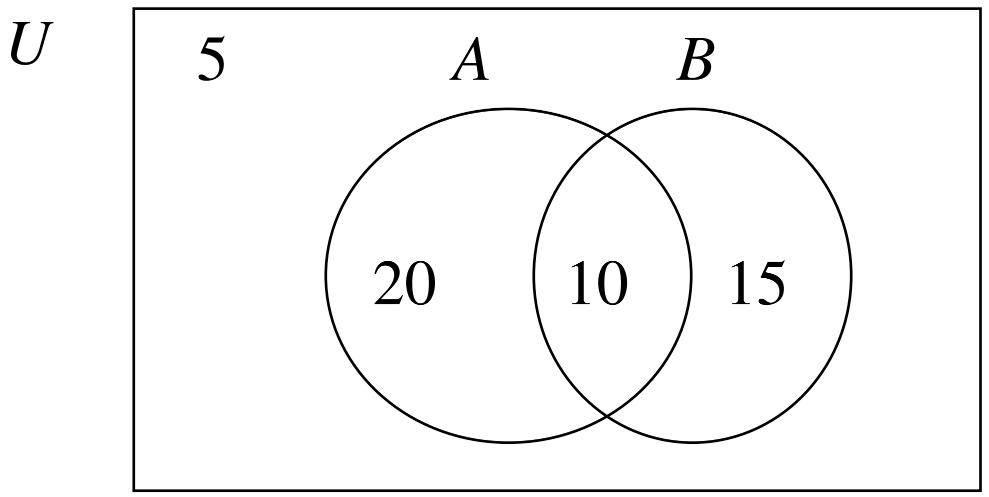
$$|\{\{1,2,3\},\{4,5\}\}| = 2$$

Universal Set

A set which has all the elements in the universe of discourse is called a *universal set*. We will usually denote this set by *U*.

Venn Diagrams

In a class of 50 college freshmen, 30 are studying Python, 25 studying C++, and 10 are studying both. How many freshmen are studying either computer language?

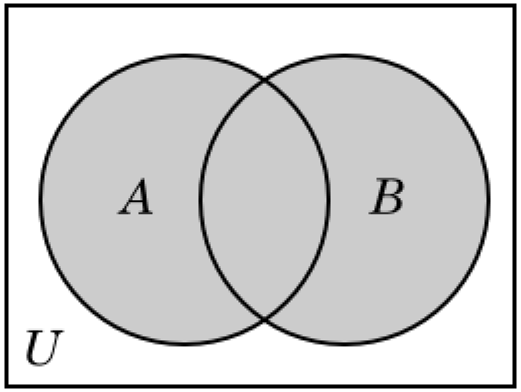


$$|A \cup B| = |A| + |B| - |A \cap B|$$

$$= 30 + 25 - 10 = 45$$

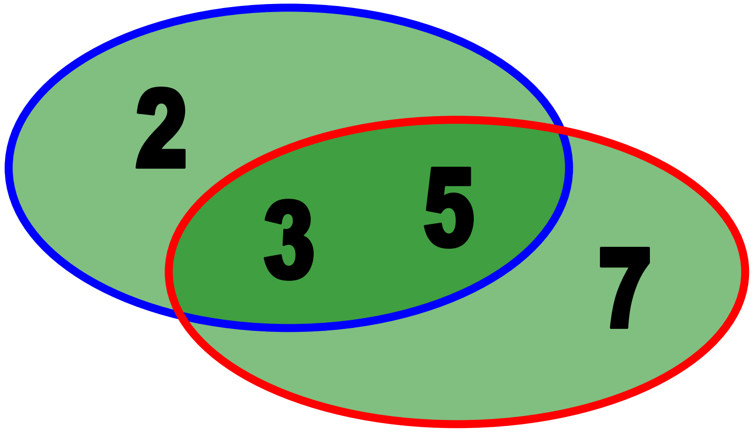
The Union Operator

- For sets A, B, their *union* $A \cup B$ is the set containing all elements that are either in A **or** in B (or in both).
- Formally, $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$.
- Note that $A \cup B$ contains all the elements of A **and** it contains all the elements of B:



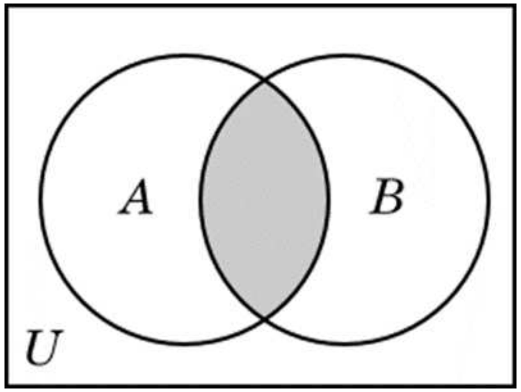
Union Examples

- $\{a,b,c\} \cup \{2,3\} = \{a,b,c,2,3\}$
- $\{2,3,5\} \cup \{3,5,7\} = \{2,3,5,3,5,7\} = \{2,3,5,7\}$



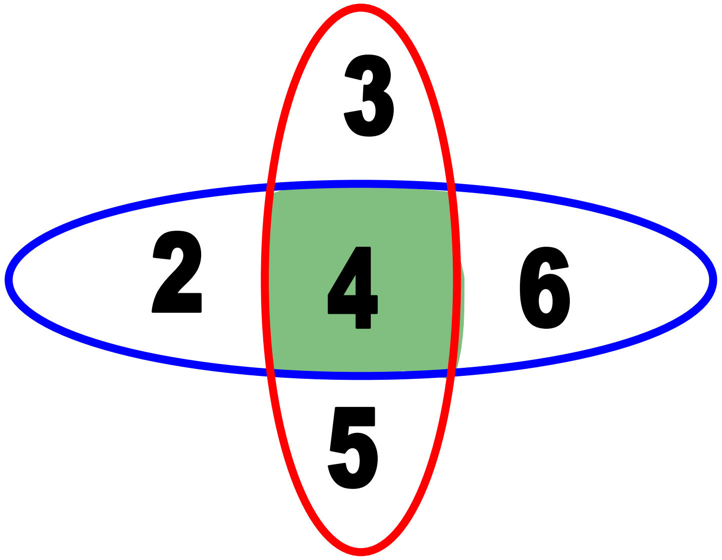
The Intersection Operator

- For sets A, B, their *intersection* $A\cap B$ is the set containing all elements that are simultaneously in A **and** in B.
- $A\cap B=\{x \mid x\in A \text{ and } x\in B\}$.
- Note that $A\cap B$ is a subset of A **and** it is a subset of B:



Intersection Examples

- $\{a,b,c\} \cap \{2,3\} = \underline{\ \emptyset\ }$
- $\{2,4,6\} \cap \{3,4,5\} = \underline{\ \{4\}\ }$



Disjointedness

- Two sets *A*, *B* are called *disjoint* (*i.e.*, unjoined) iff their intersection is empty. ($A \cap B = \varnothing$)
- Example: the set of even integers is disjoint with the set of odd integers.

Set Difference

- For sets *A*, *B*, the ***difference of A and B***, written $A - B$ or alternatively $A \setminus B$ is the set of all elements that are in *A* but not *B*.
- $A - B = \{x \mid x \in A \text{ and } x \notin B\}$
- Also called:
  The ***complement of B with respect to A***.

Set Difference Examples

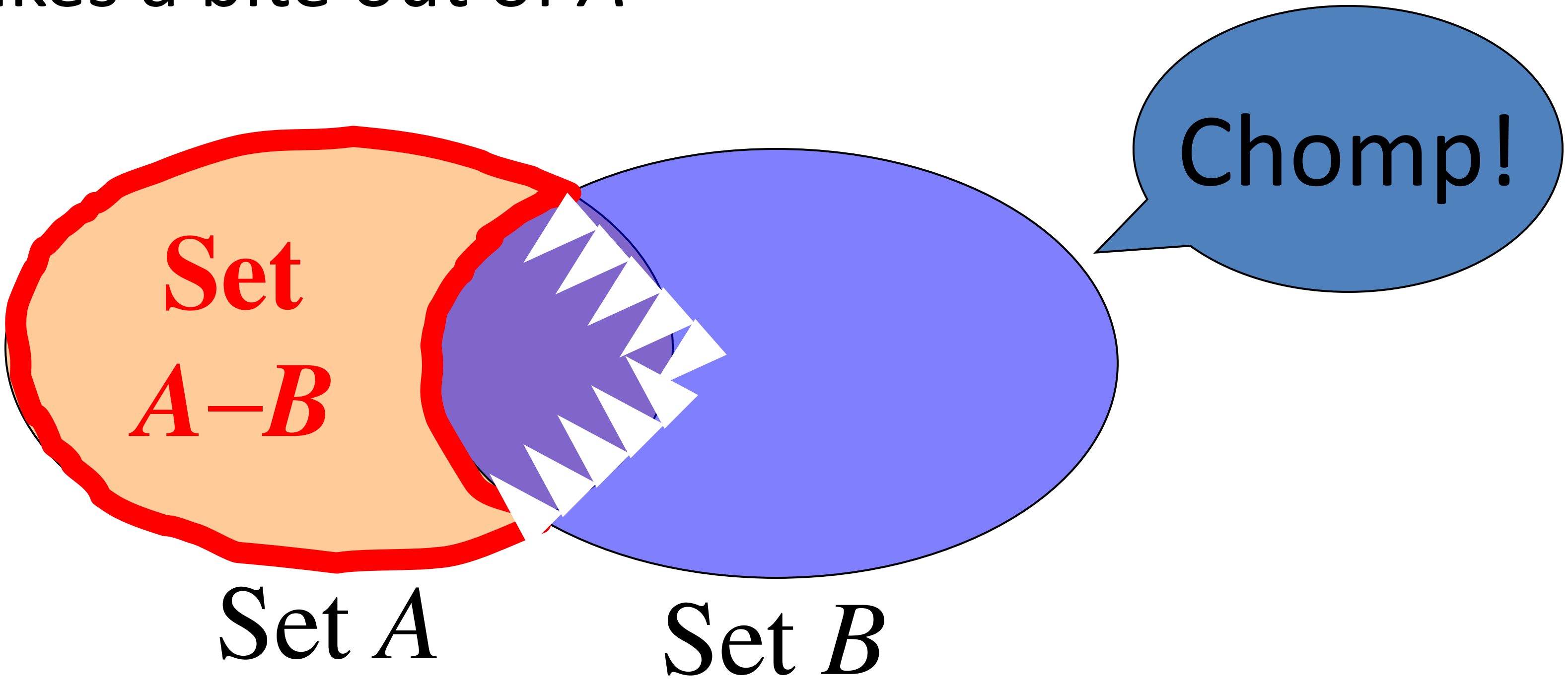
- $\{\boxed{1},\cancel{2},\cancel{3},\boxed{4},\cancel{5},\boxed{6}\} - \{2,3,5,7,9,11\} =$

  $\underline{\quad\{1,4,6\}\quad}$

- **Z** – **N** = {… , -1, 0, 1, 2, … } – {0, 1, … }
  = {*x* | *x* is an integer but not a nat. #}
  = {*x* | *x* is a negative integer}
  = {… , -3, -2, -1}

Set Difference - Venn Diagram

- *A* – *B* is what's left after *B* "takes a bite out of *A*"



Set Complements

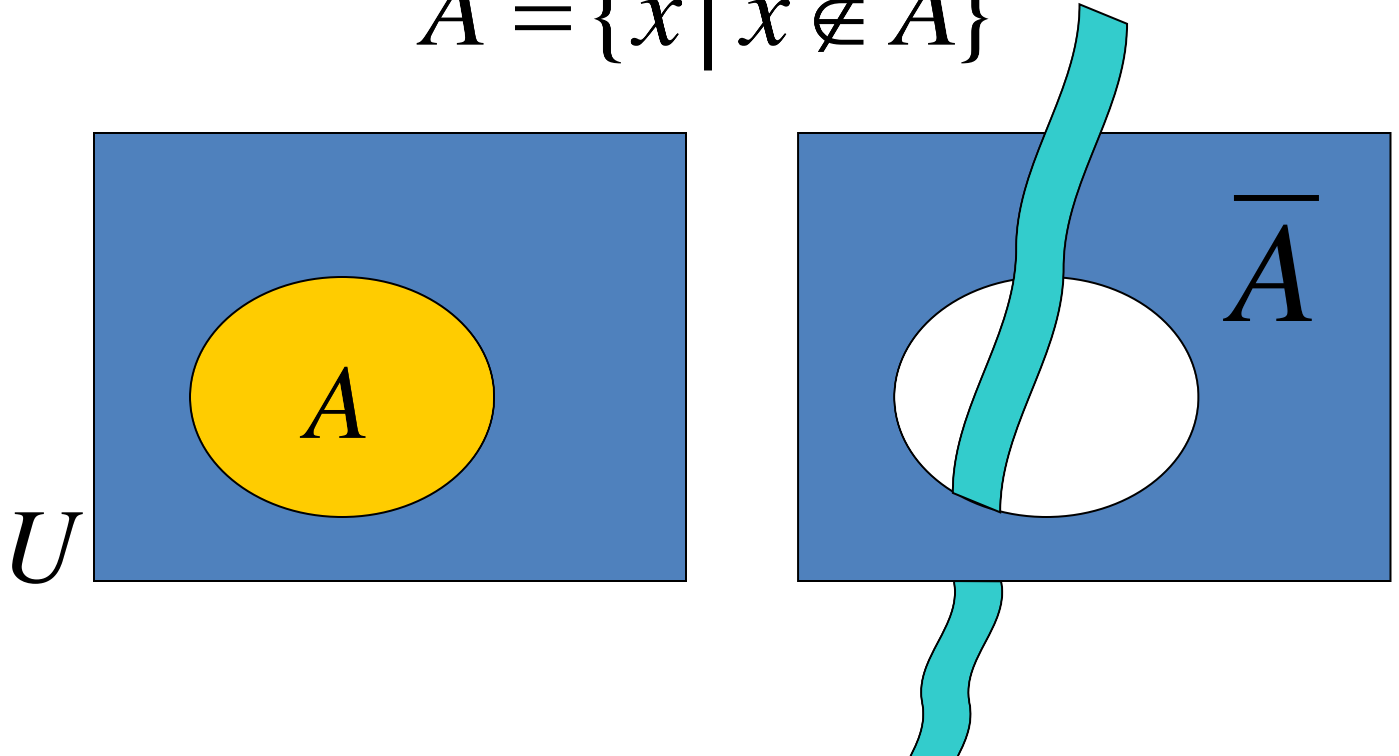
- The *universe of discourse* can itself be considered a set, call it *U*.
- The *complement* of *A*, written $\bar{A}$ or alternatively A^c, is the complement of *A* with respect to *U*, *i.e.*, it is $U - A$.
- *E.g.*, If $U = \{0,1,2,3,4,5,6,7\}$

$$\overline{\{3,5\}} = \{0,1,2,4,6,7\}$$

More on Set Complements

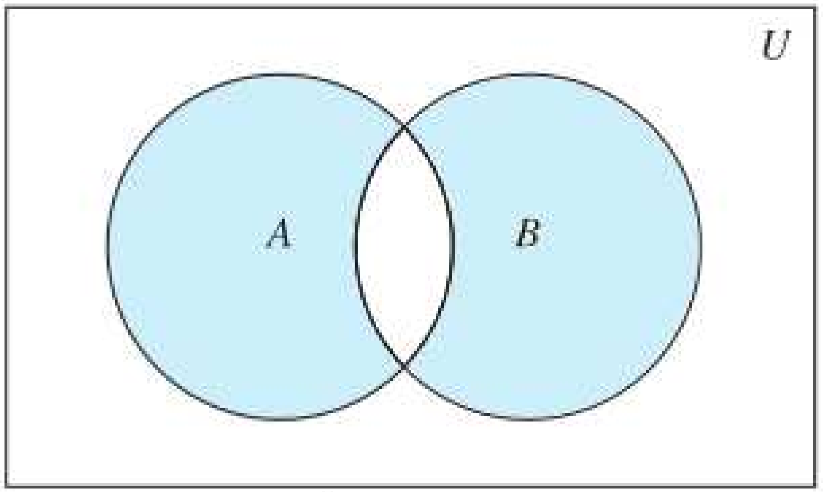
- An equivalent definition, when *U* is clear:

$$\overline{A} = \{x \mid x \notin A\}$$



The Symmetric Difference Operator

- For sets *A*, *B*, their *symmetric difference* $A \oplus B$ is the set containing all elements that are either in exactly one of the sets *A and B*
- $A \oplus B = A \cup B - A \cap B$



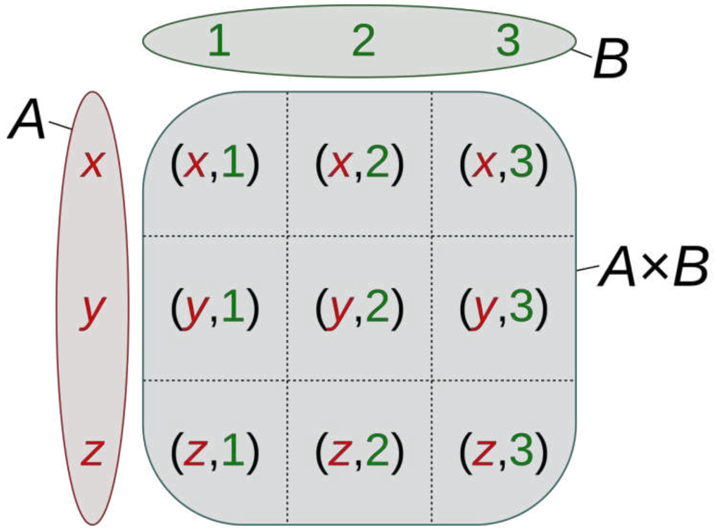
$A \oplus B$ is shaded.

Cartesian Product

- The Cartesian product of two sets A and B, denoted $A \times B$, is the set of all ordered pairs (a, b) where a is in A and b is in B.
- $A \times B = \{ (a,b) \mid a \in A \text{ and } b \in B \}$.
- $|A \times B| = |A| \times |B|$

Table representing Cartesian product

A table can be created by taking the Cartesian product of a set of rows and a set of columns. If the Cartesian product rows × columns is taken, the cells of the table contain ordered pairs of the form (row value, column value).



PSN. Pause video and work on solving

$$U = \{0,1,2,3,4,5,6,7,8,9\}, A = \{0,1,2\}, B = \{0,2,4,6\}$$

$|A| =$

$A \cup B =$

$A \cap B =$

$A - B =$

$A \oplus B =$

$A \times B =$

$|A \times B| =$

$\bar{A}$

Generalized Union

- Binary union operator: $A \cup B$
- *n*-ary union:
  $A \cup A_2 \cup \ldots \cup A_n$
  (grouping & order is irrelevant because $\cup$ is commutative and associative)
- "Big U" notation: $\bigcup_{i=1}^{n} A_i$
- Or for infinite sets of sets: $\bigcup_{A \in X} A$

Generalized Intersection

- Binary intersection operator: $A \cap B$
- n-ary intersection:
  $A_1 \cap A_2 \cap \ldots \cap A_n$
  (grouping & order is irrelevant because $\cap$ is commutative and associative)
- "Big Arch" notation: $\bigcap_{i=1}^{n} A_i$
- Or for infinite sets of sets: $\bigcap_{A \in X} A$

Infinite Sets ∞

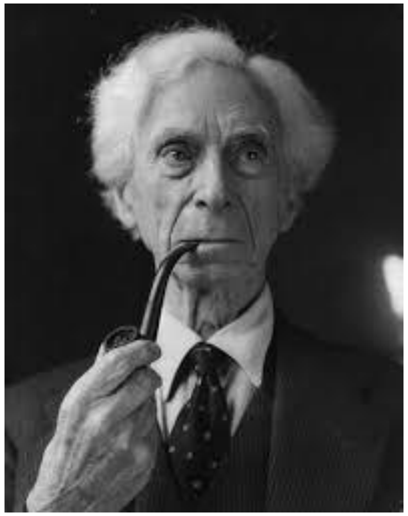
- Conceptually, sets may be *infinite* (*i.e.,* not *finite*, without end, unending).
- Symbols for some special infinite sets:
  N = {0, 1, 2, …}   The natural numbers.
  Z = {…, -2, -1, 0, 1, 2, …} The integers.
  R = The “real” numbers, such as
  374.1828471929498181917281943125…
- Is it possible to label real numbers **R** with natural numbers?

Problem Solving Notebook (PSN): Russell's Paradox

Let *R* be the set of all sets that do not contain themselves, i.e., are not members of themselves.

PSN. What is meant by a paradox?

And, why does this lead to a paradox.



Bertrand Russell

Example of Russell-Like Paradox

In a small town, where every man is clean-shaven, there is a barber.

This barber shaves all men who do not shave themselves and only men who do not shave themselves.

Who, then, shaves the barber?

-Bertrand Russell

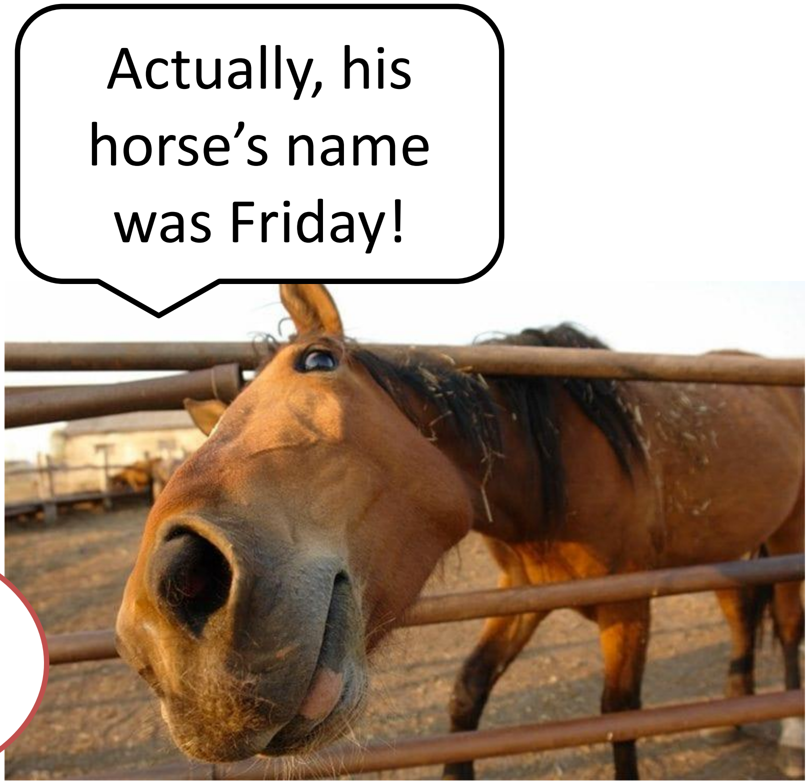
Russell’s Paradox a Milestone in Set Theory

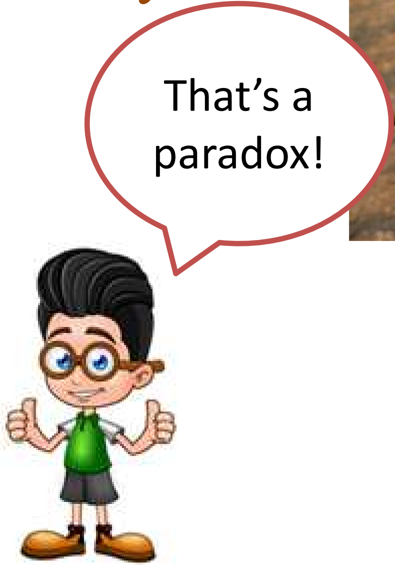
Russell's paradox threatened the foundations of mathematics. This motivated a great deal of research around the turn of the 20th century to develop a consistent (contradiction free) set theory.

In the old days when UC was just a ranch, a cowboy rode in on Friday, stayed overnight at the Bearcats Hotel, and rode out the next day on Monday. How is that possible?

Answer

His horse's name was Monday!





De Morgan’s Laws, Proving Set Identities, Infinite Cardinalities

Textbook Reading. Continue reading Chapter 1,

pages 20-25.

Set Identities

- Identity: $A \cup \emptyset = A \quad A \cap U = A$
- Domination: $A \cup U = U \quad A \cap \emptyset = \emptyset$
- Idempotent: $A \cup A = A = A \cap A$
- Double complement: $\overline{(\overline{A})} = A$
- Commutative: $A \cup B = B \cup A$

  $A \cap B = B \cap A$

  $A \oplus B = B \oplus A$
- Associative: $A \cup (B \cup C) = (A \cup B) \cup C$

  $A \cap (B \cap C) = (A \cap B) \cap C$

  $A \oplus (B \oplus C) = (A \oplus B) \oplus C$

DeMorgan's Laws for Sets

$$\overline{A \cup B} = \overline{A} \cap \overline{B}$$

$$\overline{A \cap B} = \overline{A} \cup \overline{B}$$

Proving Set Identities

To prove statements about sets, of the form $E_1 = E_2$ (where *E*'s are set expressions), here are two useful techniques:

- Prove $E_1 \subseteq E_2$ and $E_2 \subseteq E_1$ (*mutual subsets*)
  - Prove $x \in E_1 \rightarrow x \in E_2$ and $x \in E_2 \rightarrow x \in E_1$
- Use a *membership table*.

Method 1: Mutual subsets

Example: Show distributive law holds:

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C).$$

Show that $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$

- Assume $x \in A \cap (B \cup C)$ & show $x \in (A \cap B) \cup (A \cap C)$.
- We know that $x \in A$, and either $x \in B$ or $x \in C$.
  - Case 1: $x \in B$. Then $x \in A \cap B$, so $x \in (A \cap B) \cup (A \cap C)$.
  - Case 2: $x \in C$. Then $x \in A \cap C$, so $x \in (A \cap B) \cup (A \cap C)$.
- Therefore, $x \in (A \cap B) \cup (A \cap C)$.

We have shown that $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$.

Proof cont'd

Show that $(A\cap B)\cup(A\cap C) \subseteq A\cap(B\cup C)$

- Assume $x\in(A\cap B)\cup(A\cap C)$ and show $x\in A\cap(B\cup C)$.
- We know that $x\in A\cap B$ or $x\in A\cap C$
  - Case 1: $x\in A\cap B$. Then $x\in A\cap(B\cup C)$
  - Case 2: $x\in A\cap C$. Then $x\in A\cap(B\cup C)$

This shows that $(A\cap B)\cup(A\cap C) \subseteq A\cap(B\cup C)$.

We have shown that $A\cap(B\cup C)\subseteq(A\cap B)\cup(A\cap C)$ and $(A\cap B)\cup(A\cap C) \subseteq A\cap(B\cup C)$. Therefore, we have proven that

$$(A\cap B)\cup(A\cap C) = A\cap(B\cup C).$$

Method 2: Membership Tables

- Just like truth tables for propositional logic, which we’ll see later in this course.
- Columns for different set expressions.
- Rows for all combinations of memberships in constituent sets.
- Use “1” to indicate membership in the derived set, “0” for non-membership.
- Prove equivalence with identical columns.

Membership Table for Operations

A	B	$A \cup B$	$A \cap B$	$A - B$
0	0	0	0	0
0	1	1	0	0
1	0	1	0	1
1	1	1	1	0

PSN. Give Membership Tables for operations of Symmetric Difference and Complement.

Membership Table Example

Prove $(A \cup B) - B = A - B$.

A	B	$A \cup B$	$(A \cup B) - B$	$A - B$
0	0	0	0	0
0	1	1	0	0
1	0	1	1	1
1	1	1	0	0

Alternate proof of distributive law

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$

using a membership table

A B C	$A \cap B$	$A \cap C$	$B \cup C$	$A \cap (B \cup C)$	$(A \cap B) \cup (A \cap C)$
0 0 0	0	0	0	0	0
0 0 1	0	0	1	0	0
0 1 0	0	0	1	0	0
0 1 1	0	0	1	0	0
1 0 0	0	0	0	0	0
1 0 1	0	1	1	1	1
1 1 0	1	0	1	1	1
1 1 1	1	1	1	1	1

PSN. Prove $(A \cup B) - C = (A - C) \cup (B - C)$ using a membership table.

The Power Set Operation

- The *power set P*(*S*) of a set *S* is the set of all subsets of *S*. $P(S) = \{x \mid x \subseteq S\}$.
- *E*.*g*. $P(\{a,b\}) = \{\emptyset, \{a\}, \{b\}, \{a,b\}\}$.
- Sometimes $P(S)$ is written 2^S.
  Note that for finite *S*, $|P(S)| = 2^{|S|}$.
- It is easily show that for finite sets |P(S)| > |S| (where N denote the natural numbers).
- For infinite sets *S* is also true that the cardinality of *P*(*S*) is strictly greater than the cardinality of *S*. This can be shown using a generalization of Cantor's diagonal, but is beyond the scope of this course.

PSN. Pause video and work on solving

$U = \{0,1,2,3,4,5,6,7,8,9\}, A = \{0,1,2\}, B = \{0,2,4,6\}$

$P(B) =$

$|P(A \cup B)| =$

$|P(A \times B)/ =$

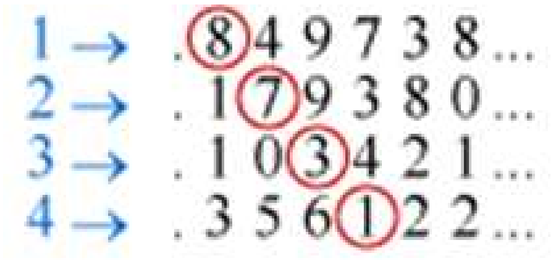
$|\ P(A) \times P(B)/ =$

Infinite cardinalities

- Cardinality of real numbers is strictly greater than natural numbers.
- In particular, you can’t list the set of real numbers between 0 and 1, or equivalently index them with the natural numbers.
- Cantor showed that at least one real number will not be included in the list, so that the subset of real number between 0 and 1 is strictly greater than the natural numbers.

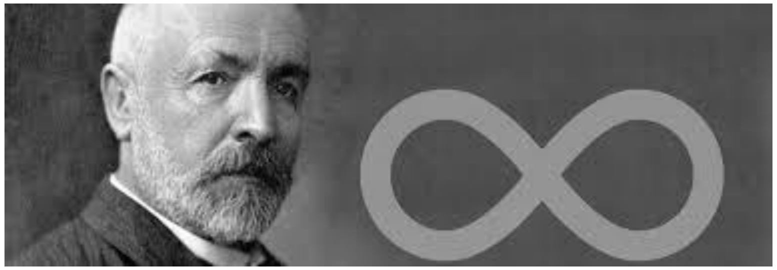
Cantor’s Diagonal

Imagine listing all real numbers between 0 and 1 in any order. You can always make an unlisted real number by changing every digit on the diagonal.



$$\begin{array}{ll}
1 \rightarrow & .\textcircled{8}\,4\,9\,7\,3\,8\ldots \\
2 \rightarrow & .\,1\,\textcircled{7}\,9\,3\,8\,0\ldots \\
3 \rightarrow & .\,1\,0\,\textcircled{3}\,4\,2\,1\ldots \\
4 \rightarrow & .\,3\,5\,6\,\textcircled{1}\,2\,2\ldots
\end{array}$$

Georg Cantor

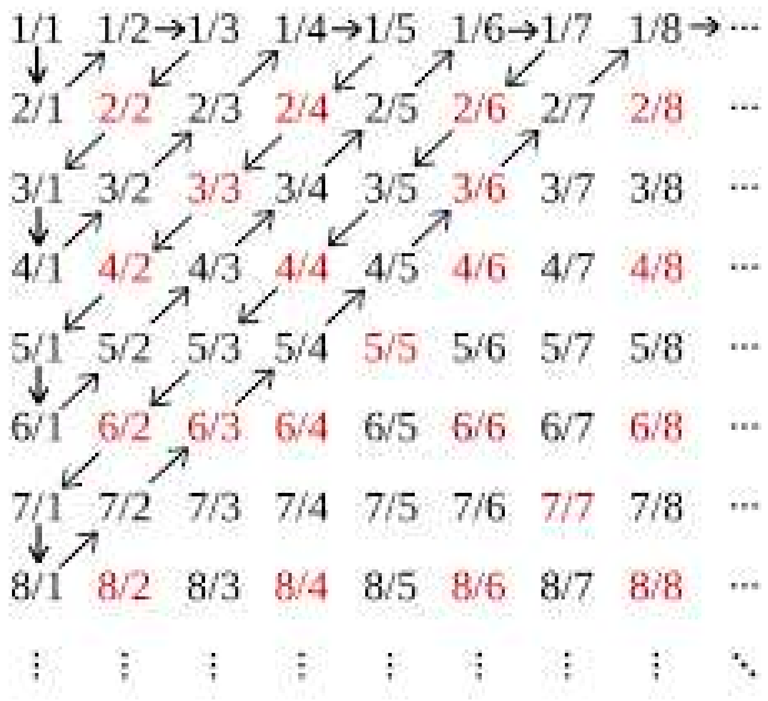


Countable

- We say the cardinality of the natural numbers is countable.
- The cardinality of the real numbers is uncountable.
- A countable set is a set with the same cardinality as some subset of the set of natural numbers. A countable set is either a finite set or a countably infinite set.

Rational numbers are countable

Diagram below shows how to list all the rational numbers by following the arrows. Numbers in red indicated rational numbers that have already been counted and need not be added to the list.



Meaning of Discrete

- Discrete mathematics deals with structures that are finite or infinite but countable.
- A random variable is said to be discrete if the set of values it can take is either finite or infinite but countable.

Continuum Hypothesis

- In mathematics, the continuum hypothesis (abbreviated CH) states:

  There is no set whose cardinality is strictly between that of the integers and the real numbers.

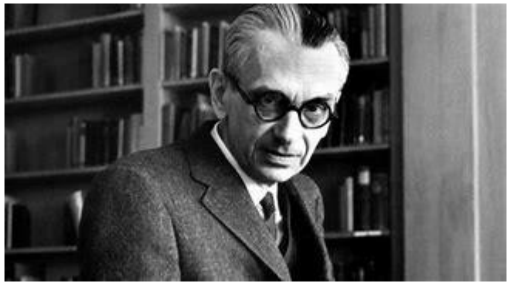
- Despite his efforts Cantor could not resolve CH.

Continuum Hypothesis

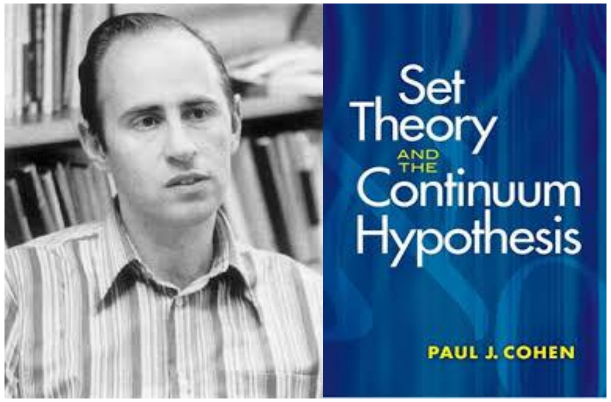
- The problem persisted and was considered so important by Hilbert that he placed it first on his famous list of open problems to be faced by the 20th century. Hilbert also struggled to resolve CH, again without success.

David Hilbert

- Ultimately, this lack of progress was explained by the combined results of Gödel and Cohen, which together showed that CH cannot be resolved on the basis of the axioms that mathematicians were employing.



Kurt Friedrich Gödel



Joke (bad)

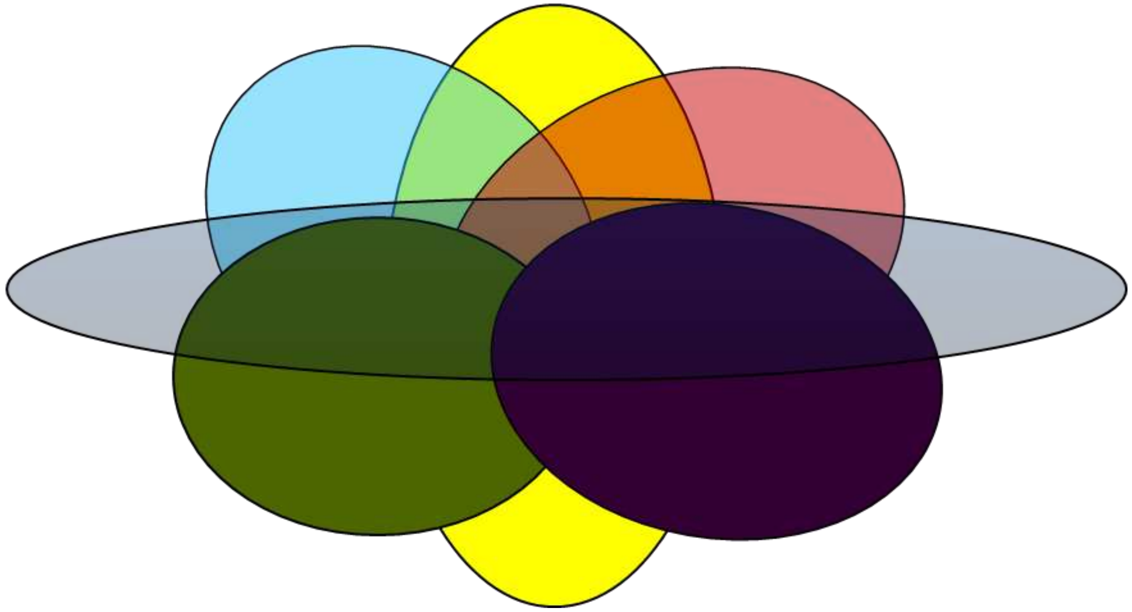
Why did the engineering students not finish the lecture video?

They were getting a little ANSI.

American National Standards Institute

This joke is not only bad, it's nerdish bad!

Principle of Inclusion-Exclusion



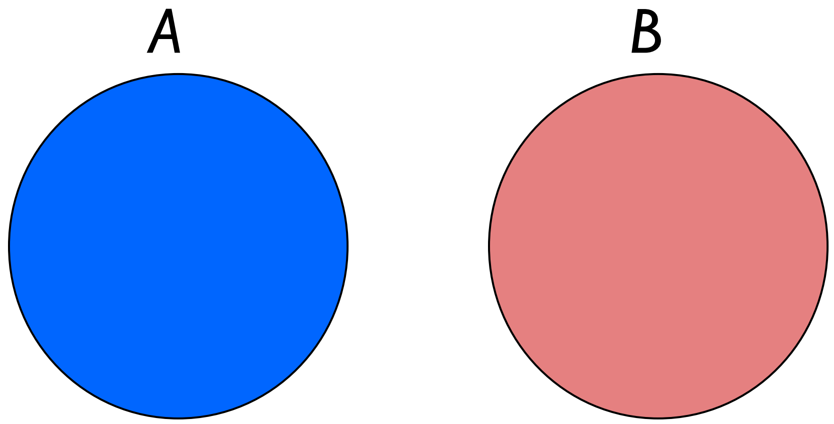
Textbook Reading:

Section 1.5, pp. 34-41

Sum Rule

If sets A and B are disjoint, then

$$|A \cup B| = |A| + |B|$$

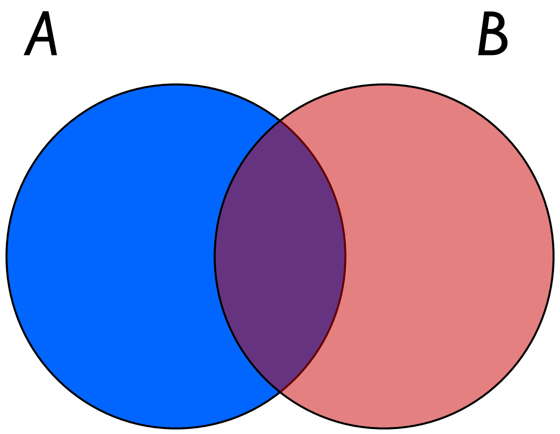


What if A and B are not disjoint?

Inclusion-Exclusion (2 sets)

For two arbitrary sets *A* and *B*

$$|A \cup B| = |A| + |B| - |A \cap B|$$



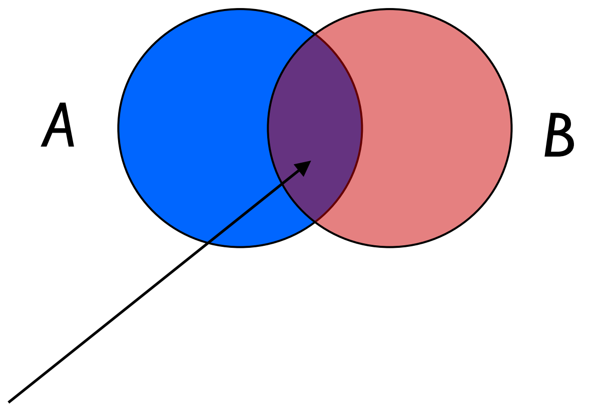
Example Inclusion-Exclusion (2 sets)
How many numbers from 1 to 1000 are multiples of 3 or 5

Let S be the set of integers from 1 to 1000 that are multiples of 3 or multiples of 5.

Let A be the set of integers from 1 to 1000 that are multiples of 3.

Let B be the set of integers from 1 to 1000 that are multiples of 5.

It is clear that S is the union of A and B,
but notice that A and B are not disjoint.



|A| = 1000/3 = 333 |B| = 1000/5 = 200

A ∩ B is the set of integers that are multiples of 15, and so |A ∩ B| = 1000/15 = 66

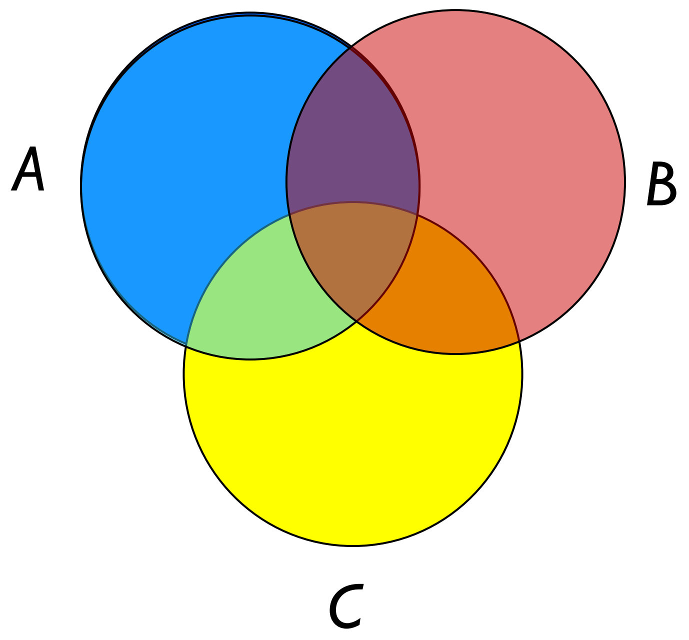
So, by the inclusion-exclusion principle, we have |S| = |A| + |B| – |A ∩ B| = 467.

Sometimes it is useful to apply the Principle of Optimality to the **complement** of a set. For example, suppose A is the set of numbers between 1 and n that are relatively prime to a n. To compute $|A|$ it is easier to first compute $|\bar{A}|$ using the Principle of Optimality, then computing $|A|$ by $|A| = n - |\bar{A}|$.

PSN. Let p and q be any two prime numbers and let $n = pq$. Applying Principle of Inclusion-Exclusion and complement sets show that the number of positive numbers less than n that are relatively prime to n (i.e., have no factor greater than 1 in common with n is $(p - 1)(q - 1)$.

This has application to the important RSA cryptosystem, which we will discuss later in the course.

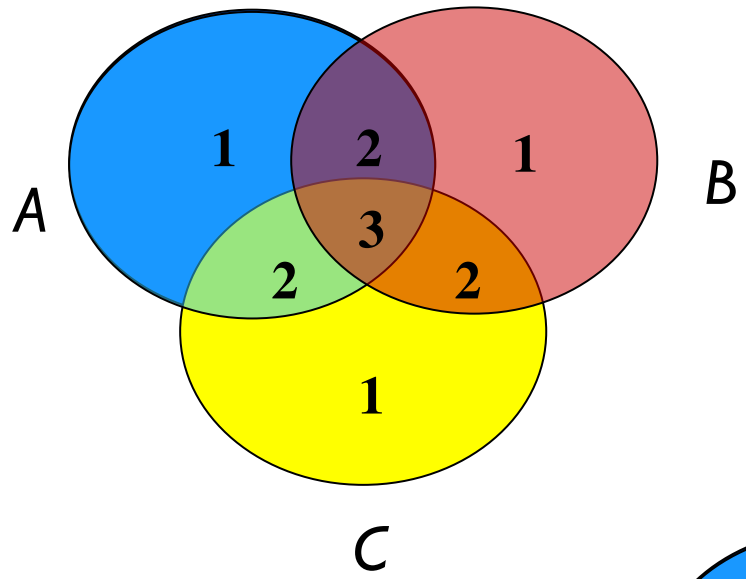
Inclusion-Exclusion (3 sets)



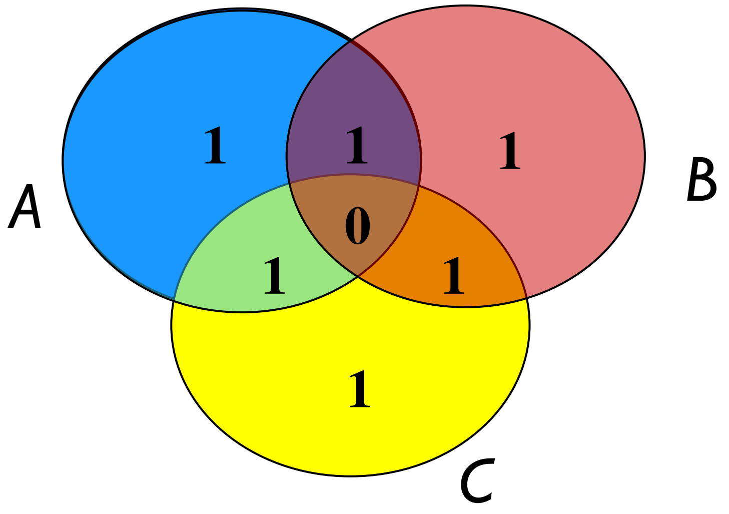
$$\boxed{\begin{aligned}|A \cup B \cup C| = |A| + |B| + |C| \\ - |A \cap B| - |A \cap C| - |B \cap C| \\ + |A \cap B \cap C|\end{aligned}}$$

Proof Inclusion-Exclusion (3 sets)

$|A| + |B| + |C|$



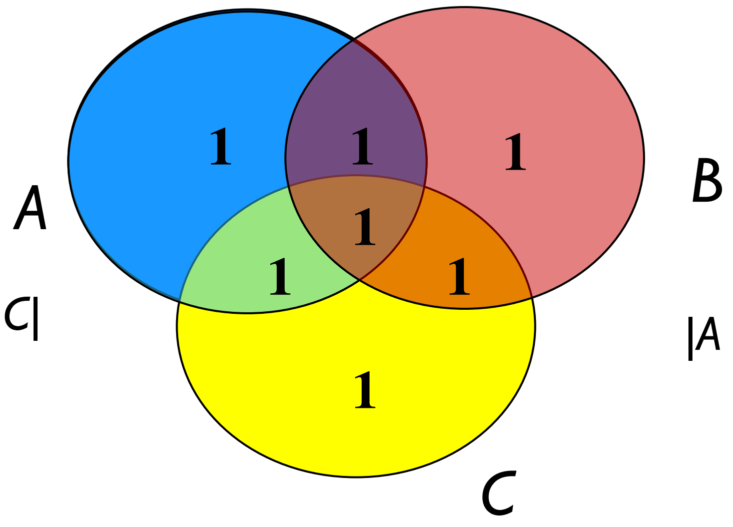
$|A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C|$



$|A| + |B| + |C|$

$- |A \cap B| - |A \cap C| - |B \cap C|$

$+ |A \cap B \cap C|$



$|A \cup B \cup C|$

Inclusion-Exclusion (3 sets)

From a total of 50 students:

$50 - |A \cup B \cup C| = 3$

How many know none?

How many know all?

$|A \cap B \cap C|$

$\lvert A\rvert \longrightarrow$	30 know Java
$\lvert B\rvert \longrightarrow$	18 know C++
$\lvert C\rvert \longrightarrow$	26 know C#
$\lvert A \cap B\rvert \longrightarrow$	9 know both Java and C++
$\lvert A \cap C\rvert \longrightarrow$	16 know both Java and C#
$\lvert B \cap C\rvert \longrightarrow$	8 know both C++ and C#
$\lvert A \cup B \cup C\rvert \longrightarrow$	47 know at least one language.

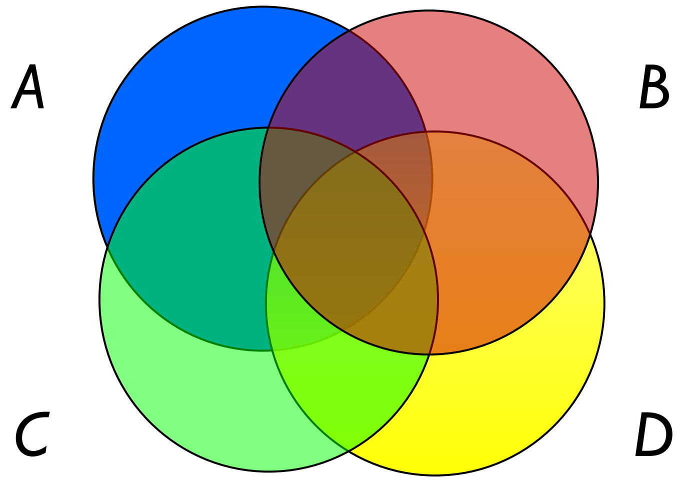
$$\boxed{|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|}$$

$47 = 30 + 18 + 26 - 9 - 16 - 8 + |A \cap B \cap C|$

$|A \cap B \cap C| = 6$

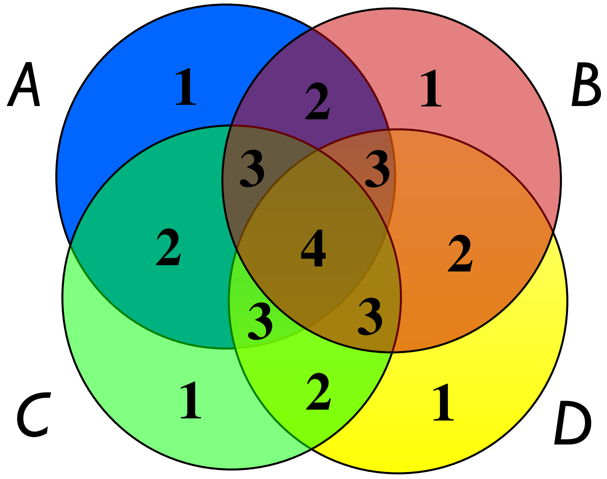
PSN. Using set complement and the Inclusion-Exclusion with 3 sets, obtain a formula for the number of numbers between 1 and 500, inclusive, that are relatively prime to 60.

Inclusion-Exclusion (4 sets)

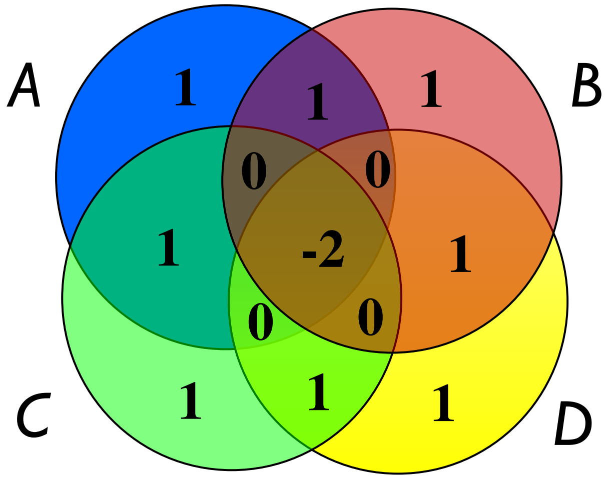


$$
\begin{aligned}
|A \cup B \cup C \cup D| = {} & |A| + |B| + |C| + |D| \\
& - |A \cap B| - |A \cap C| - |A \cap D| - |B \cap C| - |B \cap D| - |C \cap D| \\
& + |A \cap B \cap C| + |A \cap B \cap D| + |A \cap C \cap D| + |B \cap C \cap D| \\
& - |A \cap B \cap C \cap D|
\end{aligned}
$$

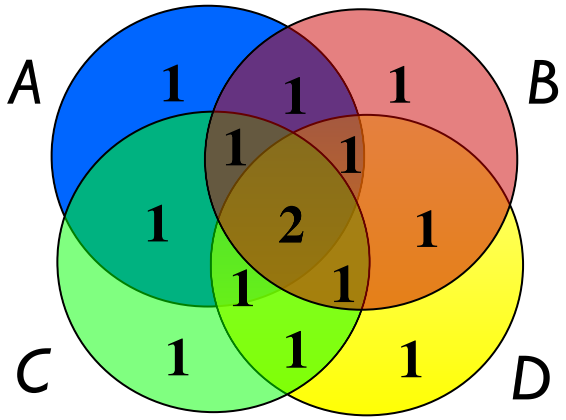
$|A| + |B| + |C| + |D|$



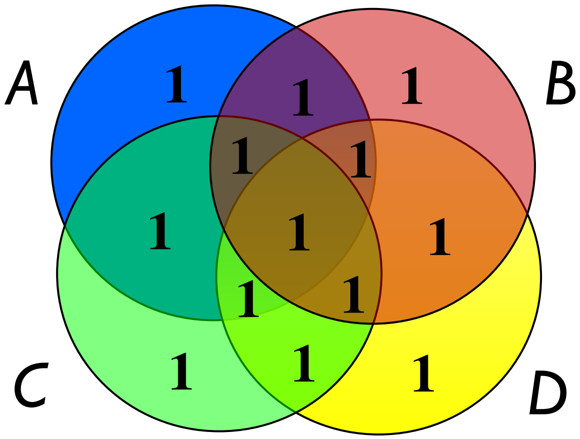
$|A| + |B| + |C| + |D|$
$- |A \cap B| - |A \cap C| - |A \cap D| - |B \cap C| - |B \cap D| - |C \cap D|$



$|A| + |B| + |C| + |D|$
$- |A \cap B| - |A \cap C| - |A \cap D| - |B \cap C| - |B \cap D| - |C \cap D|$
$+ |A \cap B \cap C| + |A \cap B \cap D| + |A \cap C \cap D| + |B \cap C \cap D|$

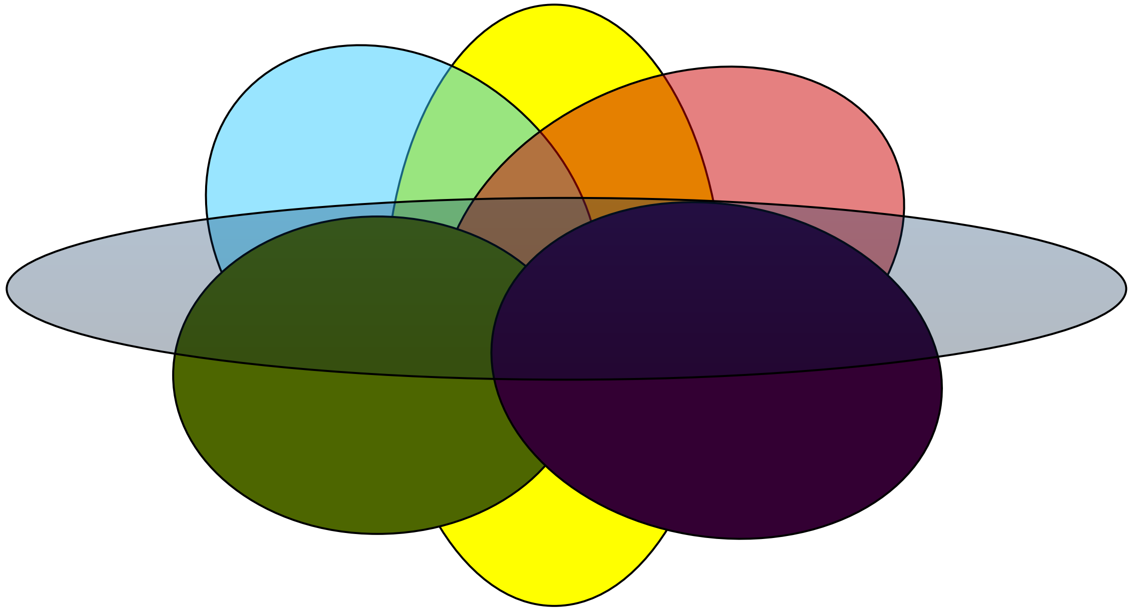


$|A| + |B| + |C| + |D|$
$- |A \cap B| - |A \cap C| - |A \cap D| - |B \cap C| - |B \cap D| - |C \cap D|$
$+ |A \cap B \cap C| + |A \cap B \cap D| + |A \cap C \cap D| + |B \cap C \cap D|$
$- |A \cap B \cap C \cap D| = |A \cup B \cup C \cup D|$



Inclusion-Exclusion (*n* sets)

What is the inclusion-exclusion formula for the union of *n* sets?



Inclusion-Exclusion (*n* sets)

$$\left|A_1 \cup A_2 \cup \cdots \cup A_n\right| =$$

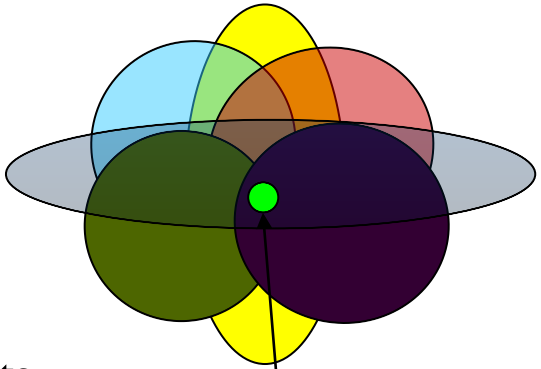
sum of sizes of all single sets
– sum of sizes of all 2-set intersections
+ sum of sizes of all 3-set intersections
– sum of sizes of all 4-set intersections
…
+ $(-1)^{n+1} \times$ sum of sizes of intersections of all *n* sets

$$= \sum_{k=1}^{n} (-1)^{k+1} \sum_{\substack{S \subseteq \{1,2,\ldots,n\} \\ |S|=k}} \left| \bigcap_{i \in S} A_i \right|$$

Inclusion-Exclusion (n sets)

$|A_1 \cup A_2 \cup A_3 \cup \ldots \cup A_n|$

sum of sizes of all single sets
– sum of sizes of all 2-set intersections
+ sum of sizes of all 3-set intersections
– sum of sizes of all 4-set intersections
…
+ $(-1)^{n+1} \times$ sum of sizes of intersections of all *n* sets



We want to show that every element is counted exactly once.

Consider an element which belongs to exactly k sets, say $A_1, A_2, A_3, \ldots, A_k$.

In the formula, such an element is counted the following number of times:

$$\binom{k}{1} - \binom{k}{2} + \binom{k}{3} - \binom{k}{4} + \ldots + (-1)^{k+1}\binom{k}{k} = 1$$

Therefore each element is counted exactly once, and thus the formula is correct

Binomial Coefficient and Binomial Theorem will be covered later in course in Chapter 7 (see page 459)

$$\binom{k}{1} - \binom{k}{2} + \binom{k}{3} - \binom{k}{4} + \ldots + (-1)^{k+1}\binom{k}{k} = 1$$

$$\boxed{(x+y)^k = \sum_{k=0}^{n} \binom{k}{i} x^i y^{k-i}}$$

Plug in *x* = 1 and *y* = -1 in the above binomial theorem, we have

$$0 = \binom{k}{0} - \binom{k}{1} + \binom{k}{2} + \ldots + (-1)^i \binom{k}{i} + \ldots + (-1)^k \binom{k}{k}$$

$$\Rightarrow \binom{k}{1} - \binom{k}{2} + \ldots + (-1)^{i+1}\binom{k}{i} + \ldots + (-1)^{k+1}\binom{k}{k} = \binom{k}{0}$$

$$= 1$$

Hope you had a good breakfast this morning.
What type of bagel can fly?

Answer:

a plain bagel

Three Standard Proof Techniques

1. Disproof by Counterexample
2. Proof by Contradiction
3. Mathematical Induction

Disproof by Counterexample

A **counterexample** is an example that disproves a statement or proposition.

Counterexamples are important because they enable mathematicians to show that certain conjectures or ideas, are false

Everyday Coin Changing

Make change of C cents using **fewest** coins by first choosing as many quarters as possible, then as many dimes, then as many nickels, and finally pennies. This can be programmed as follows using integer division, which truncates to an integer.

```
Quarters = C / 25;
R = C – 25*Quarters;
Dimes = R / 10;
R = R – 10*Dimes;
Nickels = R / 5;
Pennies = R – 5*Nickels
```

Same Algorithm without Nickels

```
Quarters = C / 25;
R = C – 25*Quarters;
Dimes = R / 10;
Pennies = R – 10*Dimes;
```

PSN. Is this algorithm still correct, i.e., are fewest coins used?

(pause video to think about this)

What is Proof by Contradiction?

Proof by contradiction establishes the truth of a proposition, by showing that **assuming the proposition to be false** leads to **a contradiction,** i.e., a contradiction to the assumption or something known to be false.

Proof by contradiction is also known as indirect proof, proof by assuming the opposite, and reduction to impossibility or absurdity.

Example

Prove using contradiction that the square root of 2 is irrational, i.e., cannot be written in the form *a*/*b* where a and b are positive integers.

Proof by contraction that square root of 2 is irrational:

Let's suppose $\sqrt{2}$ were not irrational, i.e., is a rational number. Then we can write it $\sqrt{2} =$ *a/b* where *a*,*b* are whole numbers, *b* not zero. We additionally assume that this *a/b* is simplified to the lowest terms, since that can obviously be done with any fraction. Notice that in order for *a/b* to be in its simplest terms, both a and *b* must be **not be even**. One or both must be odd. Otherwise, you could simplify.

Proof by contraction that square root of 2 is irrational:

From the equality $\sqrt{2} = a/b$ it follows that $2 = a^2/b^2$, or $a^2 = 2 * b^2$. So the square of a is an even number since it is two times something. From this we can know that a itself is also an even number. Why? Because it can't be odd; if a itself was odd, then $a * a$ would be odd too. Thus, $a = 2k$ where k is this other number.

Proof by contraction that square root of 2 is irrational:

If we substitute $a = 2k$ into the equation $2 = a^2/b^2$, we get:

$$2 = (2k)^2/b^2$$

$\rightarrow 2 = 4k^2 / b^2 \rightarrow 2b^2 = 4k^2 \rightarrow b^2 = 2k^2$

This means b^2 is even, from which follows again that *b* itself is an even number! This is a **contradiction**, because we started the whole process saying that *a/b* is simplified to the lowest terms, and now it turns out that *a* and *b* would both be even. So $\sqrt{2}$ cannot be rational.

Second Example: Coin Changing

Consider again the problem of returning (correct) change using quarters, dimes, nickels, pennies. Greedy Algorithm chooses the most quarters, then the most dimes for remaining change, etc.

Prove that Greedy Algorithm returns fewest coins using proof by contradiction.

Proof by contradiction

Assume greedy method of making change does not involve the fewest coins. Now consider an optimal solution that makes the same change C, but uses the fewest coins.

Let g_{25}, g_{10}, g_5, g_1 be the number of quarters, dimes, nickels, pennies in the greedy solution.

Let p_{25}, p_{10}, p_5, p_1 be the number of quarters, dimes, nickels, pennies in the optimal solution.

Since the greedy and optimal solution make the same change C we have

$$C = 25 \times g_{25} + 10 \times g_{10} + 5 \times g_5 + g_1 = 25 \times p_{25} + 10 \times p_{10} + 5 \times p_5 + p_1$$

Then, based on our assumption that the greedy does not involve the fewest coins, we have

$$g_{25} + g_{10} + g_5 + g_1 > p_{25} + p_{10} + p_5 + p_1$$

Trick

The clever idea (trick) in getting a handle on the proof is to make some observations about the optimal solution.

PSN. Obtain upper bounds p_{10}, p_5, p_1
(pause video to think about this)

We have shown that $p_{10} \leq 2$, $p_5 \leq 1$, $p_1 \leq 4$ and if $p_5 = 1$, then $p_{10} \leq 1$.

First consider the case where there are no nickels in the optimal solution, i.e., $p_5 = 0$. Then the most change the optimal solution can make using only dimes, nickels and pennies, involves 2 dimes and 4 pennies for a total of 24₵.

Now consider the case where there is one nickel, i.e., $p_5 = 1$. Then the most change the optimal solution can make using only dimes, nickels and pennies, involves 1 dimes, 1 nickel and 4 pennies for a total of 19₵.

Number of quarters chosen by greedy and optimal solutions

Assume the optimal and greedy solution differ in the number of quarters chosen, i.e., $g_{25} \neq p_{25}$. By definition of the greedy method it chooses more quarters, i.e., $g_{25} > p_{25}$.

Since the greedy and optimal solution make the same amount of change C, i.e.,

$$C = 25 \times g_{25} + 10 \times g_{10} + 5 \times g_5 + g_1 = 25 \times p_{25} + 10 \times p_{10} + 5 \times p_5 + p_1$$

the optimal solution needs to make up the shortage of at least 25₵ using only dimes, nickels and quarters. But, this is impossible since we showed on the previous slide that the optimal solution can make change of at most 24₵ using only dimes, nickels and pennies. Since we have obtained a contradiction, we can conclude the that greedy and optimal choose the same number of quarters, i.e., $\boldsymbol{g_{25} = p_{25}}$.

We’ve shown that $g_{25} = p_{25}$

Now consider the remaining change R after using the quarters are used, i.e.,

$$R = C - 25 \times g_{25} = C - 25 \times p_{25} .$$

Using a similar argument, we can show that the greedy and optimal solutions use the same number of dimes. Otherwise, optimal solution is short at least one dime and can make at most 9 cents using 1 nickel and 4 pennies.

Updating remaining change after dimes are used, we can show they involve the same number of nickels. Otherwise, optimal solution is short at least one nickel and can only make at most 4 cents using pennies.

All that is left is pennies and since the greedy and optimal solutions make the same total change and we have shown they use the same number of quarters, dimes and nickels, they are forced to use the same number of pennies.

It follows that greedy and optimal involves exactly the same number of coins, which is a **contradiction** to assumption greedy does not involve the fewest coins.

We have obtained a contradiction to the assumption that the greedy method does not use the fewest number of coins in making change. Therefore, the opposite is true, i.e., the greedy method uses the fewest coins to make change. This completes our proof by contradiction.

General Coin-Changing Problem is hard

We have shown that the greedy method works for US denominations.

Surprisingly, the problem with general denominations is hard.

It has been shown to be NP-hard. We will discuss NP-complete and NP-hard later in this course.

There is no known polynomial time algorithm in the worst case for solving the coin-changing problem for general denominations.

Joke that makes no sense

How do you know the mint making pennies was not shut down?

Answer: It makes no cents.

Three Standard Proof Techniques

1. Disproof by Counterexample
2. Proof by Contradiction
3. Mathematical Induction

Mathematical Induction

Mathematical induction is a powerful proof technique that is important in Computer Science. It is useful in proving the correctness of algorithms, as well as in the design and analysis of algorithms.

Formal Formulation

Suppose we have a sequence of propositions $P(1), P(2), \ldots, P(n), \ldots$ for which the following two steps have been established:

Basis step: $P(1)$ is true*

Induction (or Implication) step: *if* $P(k)$ is true for any given k,

Then $P(k + 1)$ must also be true.

Then $P(n)$ is true for all positive integers n.

The validity of the Principle of Mathematical Induction can be seen as follows. Since $P(1)$ is true, the induction step shows that $P(2)$ is true. But the truth of $P(2)$ in turn implies that $P(3)$ is true, and so forth. The induction step allows this process to continue indefinitely.

*For simplicity, here we are taking basis step with $n = 1$. Later we will look at variations of mathematical induction, where we take basis step with $n = b$ for some integer b.

Example 1

$$P(n): 1^2 + 2^2 + \ldots + n^2 = n(n+1)(2n+1)/6$$

Basis step: $1^2 = 1 = 1(1+1)\dfrac{1(1+1)(2+1)}{6}$ $(P(1)$ is true$)$.

Induction step: Assume that $P(k)$ is true for a given k, so that $1^2 + 2^2 + \ldots + k^2 = k(k + 1)(2k + 1)/6$. We must show that it would follow that $P(k + 1)$ is true, namely, that $1^2 + 2^2 + \ldots + (k + 1)^2 = (k + 1)(k + 2)(2k + 3)/6$. We have

$$\begin{aligned}
1^2 + 2^2 + \cdots + k^2 + (k+1)^2 &= (1^2 + 2^2 + \cdots + k^2) + (k+1)^2 \\
&= k(k+1)\frac{(2k+1)}{6} + (k+1)^2 \text{(since } P(k) \text{ is assumed true)} \\
&= \frac{(k+1)[k(2k+1) + 6(k+1)]}{6} = \frac{(k+1)(2k^2 + 7k + 6)}{6} \\
&= \frac{(k+1)(k+2)(2k+3)}{6},
\end{aligned}$$

and therefore $P(k + 1)$ is true.

Example 2 (using less formal notation)

Show that $1 + 3 + 5 + \ldots + 2n - 1 = n^2$.

Basis Step. $1 = 1^2$

Induction Step. Assume true result is true for *n* = *k*, i.e.,

$$1 + 3 + 5 + \ldots + 2k - 1 = k^2.$$

Now **consider the case *n* = *k* + 1**.

$$\begin{aligned} &1 + 3 + 5 + \ldots + 2k - 1 + 2k + 1 \\ &= (1 + 3 + 5 + \ldots + 2k - 1) + 2k + 1 \\ &= k^2 + 2k + 1 && \text{(by \textbf{Induction Hypothesis})} \\ &= (k + 1)^2 \end{aligned}$$

This completes the induction step and the proof.

PSN. Show that $1 + 2 + \ldots + n = n(n + 1)/2$.

Example 3 – Harmonic Series

The harmonic series is defined by $H(n) = 1 + 1/2 + . . . + 1/n$

Proposition. $H(n) \leq 1 + \ln n$

Basis Step $\ln 1 = 0 < 1 \leq 0 + 1$

Induction Step. Assume proposition is true for $n = k$, i.e., $H(k) \leq 1 + \ln k + 1$

Now consider the case $n = k + 1$. Clearly, $H(k + 1) = H(k) + 1/(k + 1)$. Therefore, by the **Induction Hypothesis** we have

$$H(k + 1) \leq (1 + \ln k) + 1/(k + 1) \leq 1 + \ln (k + 1).$$

To verify the last inequality, use the result that $\ln x \leq x - 1$, for all real numbers x, $0 < x \leq 2$, so that

$$\ln k - \ln (k + 1) = \ln(k/k+1) \leq (k/k+1) - 1 = -1/(k + 1) .$$

This completes the induction step and proof of the Proposition.

Variations of induction

1. Often the sequence of propositions starts with an index different from 1, such as 0 or in general an integer b. Then the basis step starts with this initial value b. The induction step remains the same, and the two steps together establish the truth of the propositions $P(n)$ for all n greater than or equal to this initial b.

2. Sometimes the propositions are only finite in number, $P(1), \ldots, P(l)$. Then the induction step is modified to require that $k < 1$. Of course, the conclusion then drawn is that $P(1), \ldots, P(l)$ are all true if the basis and induction steps are valid.

3. The Principle of Mathematical Induction can also be stated in the following so-called *strong form*, where the induction step is as follows:

   Induction step (strong form): For any positive integer k, *if* $P(j)$ is true for *all* positive integers $j \leq k$, *then* $P(k + 1)$ must also be true.

4. A combination of the above.

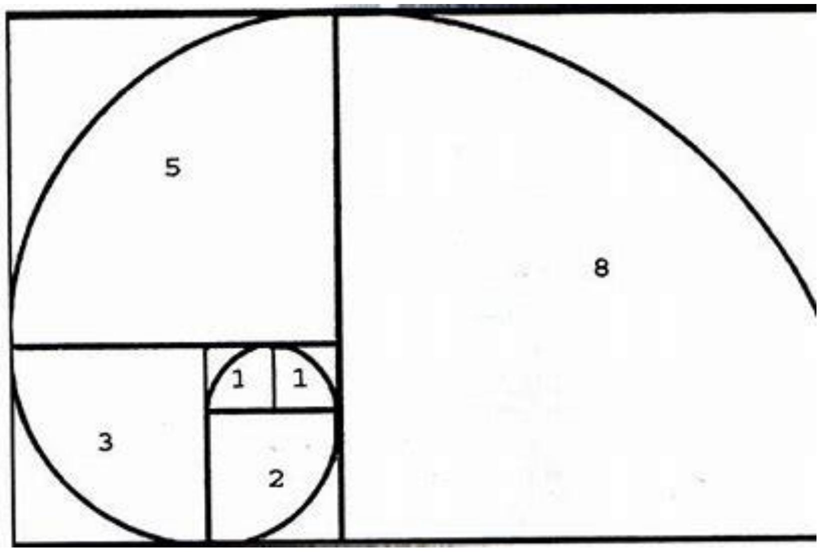
Fibonacci Numbers

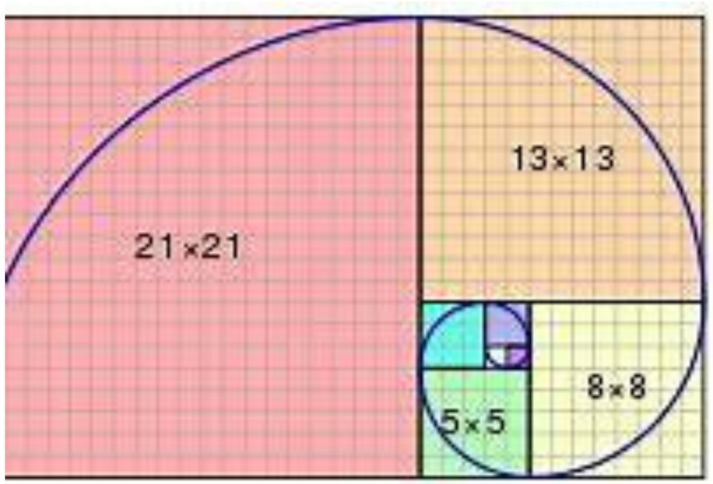
The n^{th} Fibonacci number $Fib(n)$ is defined by the recurrence relations

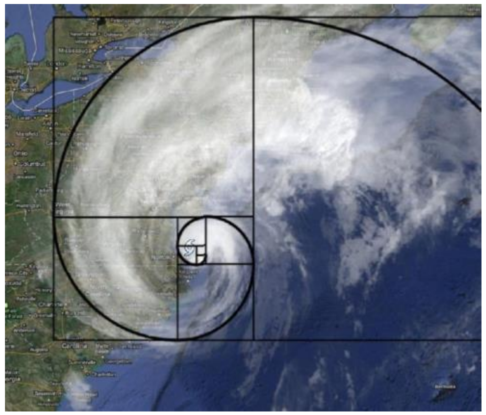
$Fib(n) = Fib(n-1) + Fib(n-2)$, $n \geq 2$, $Fib(0) = 0$, $Fib(1) = 1$.

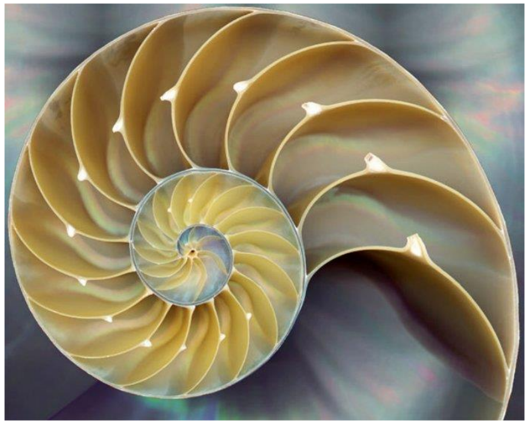
This generates sequence: 0 1 1 2 3 5 8 13 21 34 …

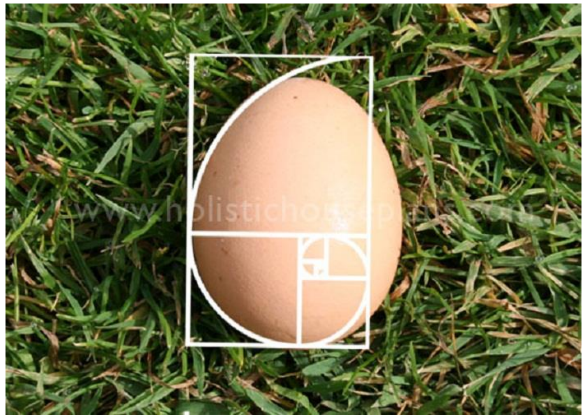
Applications in nature
Fibonacci or Golden Spiral











Example of proof using Strong Form of Induction

Proposition. $Fib(n) < 2^n$, for all $n \geq 0$.

Basis Step. Fib(0) = 0 < 2^0 and Fib(1) = 1 < 2^1. Thus, the Proposition is true for $n = 0$ and $n = 1$.

Induction Step (Strong). Assume the Proposition is true for all integers from 0 to k, i.e.,

$$Fib(n) < 2^n,\ n = 0, 1, \ldots, k.$$

Now consider the case $n = k + 1$. Using recurrence relation for Fibonacci, we obtain

$$\begin{aligned} Fib(k+1) &= Fib(k) + Fib(k-1) \\ &< 2^k + 2^{k-1} \qquad \text{(applying } \textbf{Induction Hypothesis} \text{ for } n = k \\ &\qquad\qquad\qquad\qquad \text{and } n = k-1) \\ &< 2^k + 2^k = 2^{k+1} \end{aligned}$$

This completes the induction step and the proof.

PSN. Prove $Fib(n) > 1.5^n$, for all $n \geq 11$.

Note that this is not true for any $n < 11$ (check it out), so we must start the basis step at $n = 11$.

Combining with previous result we have lower and upper bounds for $Fib(n)$:

$$1.5^n < Fib(n) < 2^n, \text{ for all } n \geq 11.$$

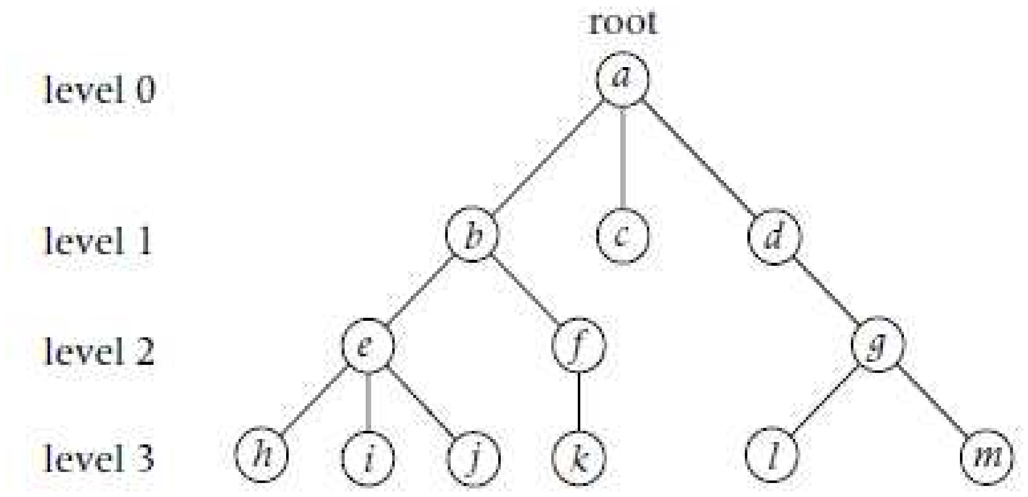
Trees

A tree is an important structure in CS with myriad applications.

- It models operations in networks such as broadcasting from a source and gathering at a sink.
- It is an important data structure used in many applications and algorithms.
- Mathematical properties of trees have important applications in the design and analysis of algorithms.

Tree Definition

A **tree** consists of a set of **nodes** (also called **vertices**), where one node is identified as the **root** and each node different from the root has another node associated with it called its **parent**. A nodes is joined to it parent using an **edge**. The set of all nodes having the same parent *p* are called the **children** of *p*. A node with no children is called a **leaf**.



Sample tree

Number of edges vs. number of nodes in a tree

Theorem. The number *m* of edges of any tree *T* is one less than the number *n* of nodes.

Proof by Induction. We perform induction on the number of *n* of nodes.

Basis Step. A tree with one node has no edges, i.e., we have $m = 0 = n - 1$

Induction Step

Assume true for $n = k$, i.e., any tree having k vertices has $k - 1$ edges.

Now consider a tree T having $n = k + 1$ nodes. For convenience, let $n(T)$ and $m(T)$ denote the number of vertices and nodes of T.

PSN. To apply the induction hypothesis, we need to perform an operation that reduces T to a tree T' having k nodes. How to do this?

Applying Induction Hypothesis

Since *T'* has *k* nodes, we can apply the induction hypothesis (inductive assumption)

$$m(T') = n(T') - 1 \quad \text{(by Induction Hypothesis)}$$

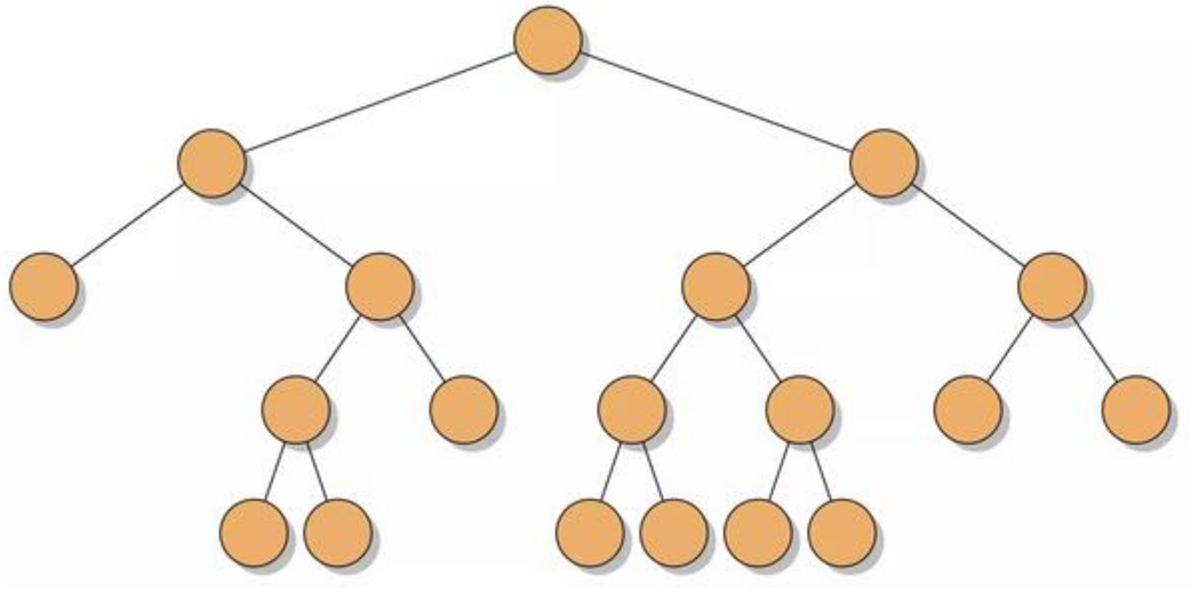
Thus, we have

$$\begin{aligned} m(T) &= m(T') + 1 \\ &= (n(T') - 1) + 1 \qquad \text{(substituting)} \\ &= n(T') = n(T) - 1. \end{aligned}$$

This completes the induction step and the proof.

2-tree

A **2-tree** is a tree where every node that is not a leaf has exactly two children.



An **internal** node is a node that is not a leaf node.

Let $I(T)$ and $L(T)$ denote the number of internal and leaf nodes of a 2-tree T, respectively. For convenience, let $I = I(T)$ and $L = L(T)$.

Proposition. Let T be a 2-tree. Then, $L = I + 1$.

Clearly, the total number n of nodes satisfies $n = I + L$, so we have:

Corollary 1. $n = 2I + 1$.

Corollary 2. $n = 2L – 1$.

Parametrizing the induction. We must decide, which parameter, we will perform induction on, i.e., the number *I* of internal nodes, the number *L* of leaf nodes or the total number *n* of nodes. We will choose *L*.

Basis Step

The proposition is true for $L = 1$. A single node 2-tree has one leaf node, the root, and 0 internal nodes, so we have

$$L = 1 = I + 1.$$

Induction Step

Assume proposition is true for $L = k$, i.e., all 2-trees T with k leaf nodes have $k - 1$ internal nodes. Now consider **any** 2-tree T having $k + 1$ leaf nodes.

PSN. To apply the induction hypothesis, we need to perform an operation that reduces T to a tree T' with k leaf nodes.

How to do this?

PSN. To verify that this construction is valid, we must prove that every 2-tree *T* contains a node, both of whose children are leaf nodes. Proof this result.

Since *T′* has one fewer leaf nodes than *T*, i.e., *T′* has *k* nodes, we can apply the induction hypothesis, i.e.,

$$L(T') = I(T') + 1.$$

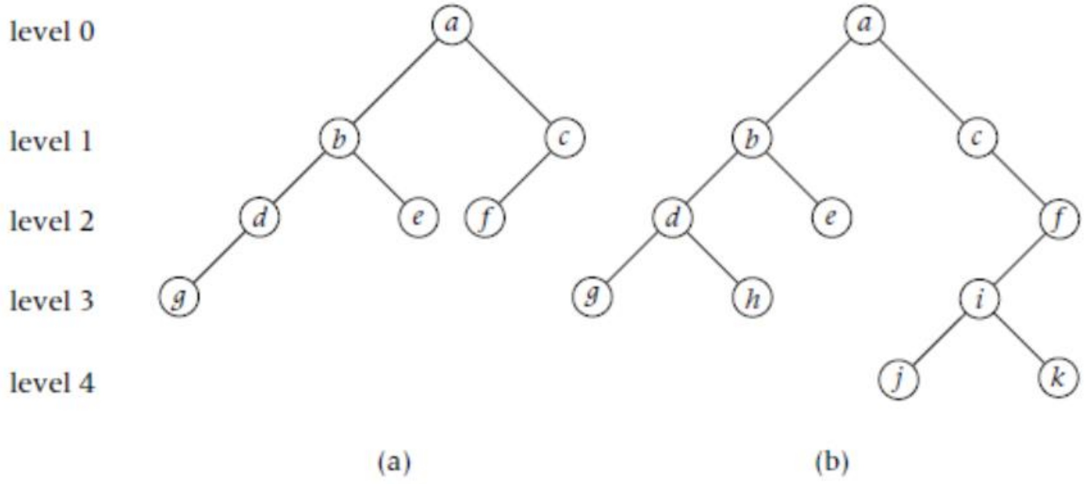
Thus,

$$L(T) = L(T') + 1 = (I(T') + 1) + 1 = I(T) + 1$$

This completes the induction step and the proof of the Proposition.

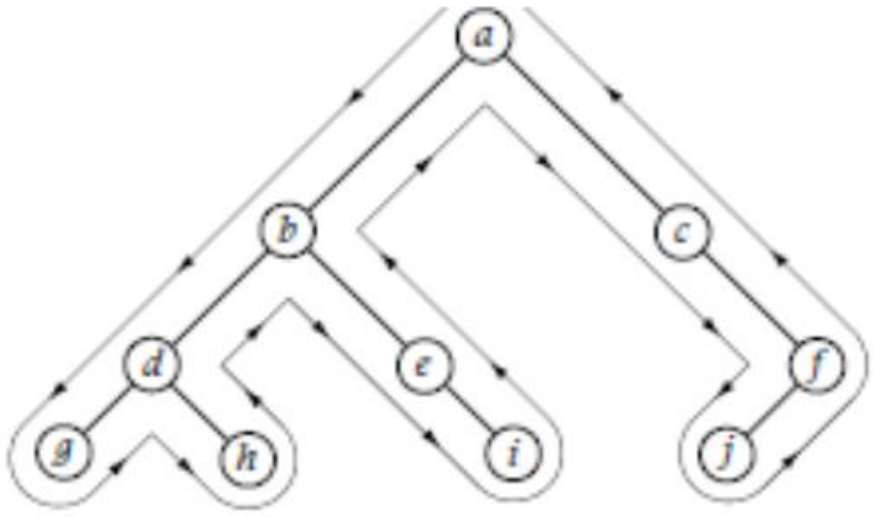
Binary Trees

A binary tree is a tree where every node has **at most** 2 children and we identify children as either a left child or right child.



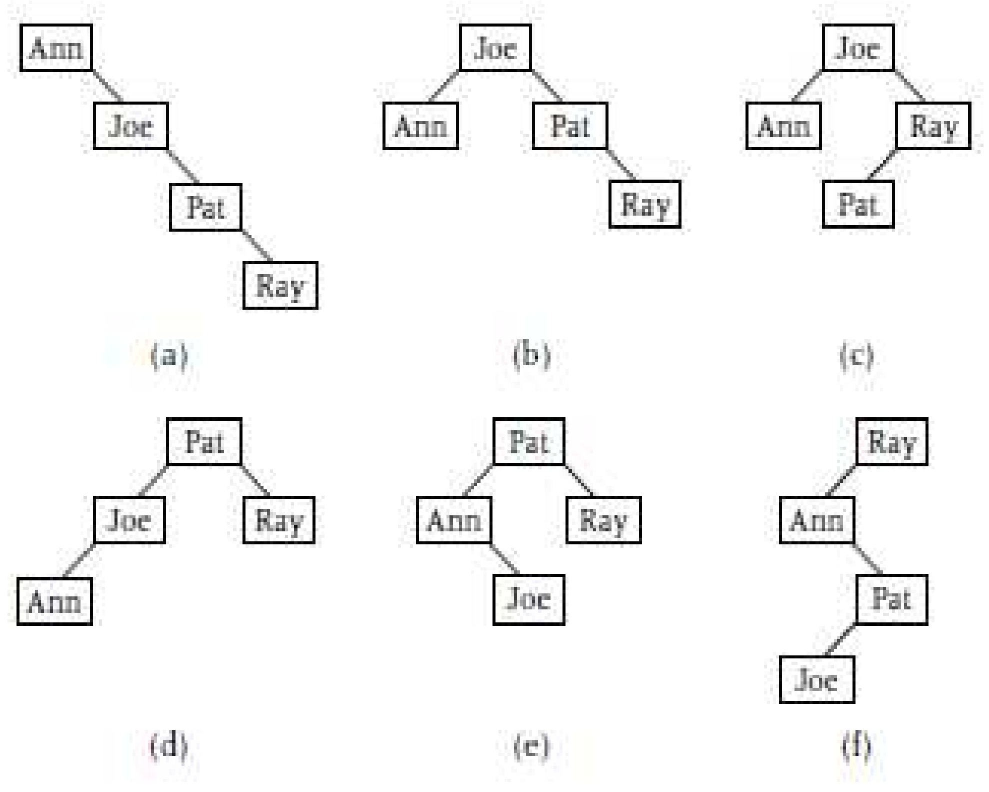
Inorder Traversal

Follow path around the tree starting at the root node and going left first when there is choice. A node with no left child is output when it is visited; otherwise the node is output the second time it is visited. The inorder traversal of the sample tree below is: gdhbeiacjf

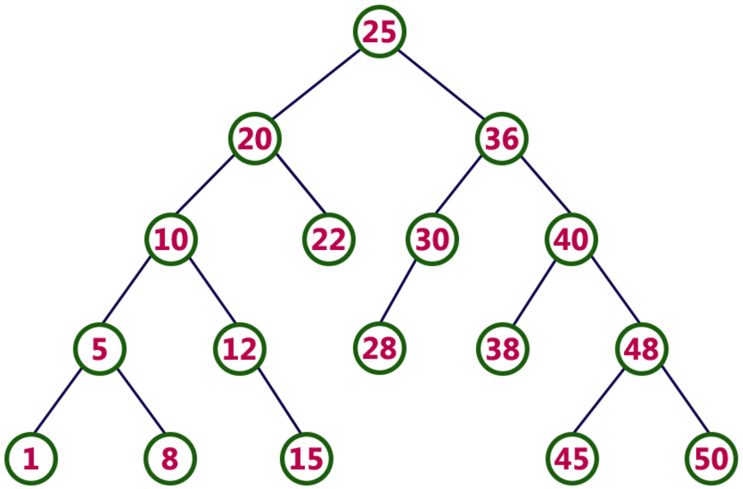


Binary Search Trees

A **binary search tree** is a binary tree with a key (value) associated with each node, so that for every node, all the keys in its left subtree are smaller and all the keys in its right subtree are larger.



Proposition. An inorder traversal of a binary search tree outputs the node keys in sorted order.



Inorder traversal: 1 5 8 10 12 15 20 22 25 28 30 36 38 40 45 48 50

Basis Step

Clearly, result is true for a binary search tree having one node.

Induction Step Strong

Assume the result is true of all binary search trees have *j* vertices, where $1 \leq j \leq k$, i.e., performing an inorder traversal of a binary search tree having *j* nodes, outputs the keys in sorted order.

Now consider a binary search tree having $k + 1$ nodes.

Applying Induction Hypothesis

- Let *L* and *R* denote the left and right subtrees of *T*.

- Since both *L* and *R* have at most *k* nodes, we can apply the **induction hypothesis** to them, i.e., **performing an inorder traversal of *L* outputs the keys in sorted order. The same applies to *R***.

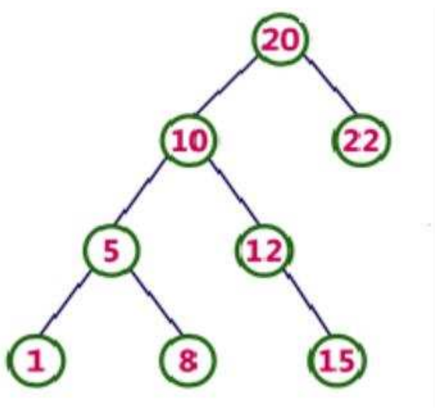
- Performing an inorder traversal of *T* involves
  - performing an inorder traversal of *L*,
  - visiting the root,
  - performing an inorder traversal of *R*.

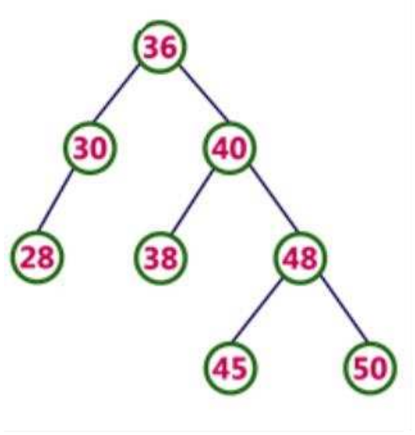
- Since all the keys in *L* are less than the root key and all the keys in *R* are greater than the root key, it follows that an inorder traversal of *T* outputs the keys in sorted order.

This completes the induction step and the proof of the Proposition.

Illustration with Previous Sample Binary Search Tree

L root *R*





1 5 8 10 12 15 20 22 25 28 30 36 38 40 45 48 50

↑ ↑ ↑

Applying Induction Hypothesis with *L*

output root key

Applying Induction Hypothesis with *R*

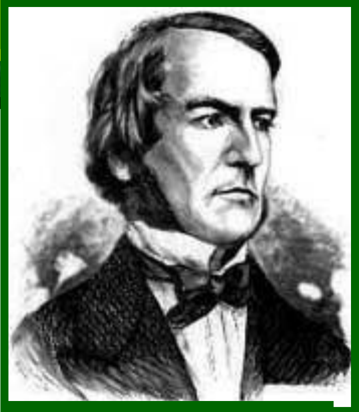
Foundations of Logic

Mathematical Logic is a tool for working with elaborate *compound* statements. It includes:

- A formal language for expressing them.
- A concise notation for writing them.
- A methodology for objectively reasoning about their truth or falsity.
- It is the foundation for expressing formal proofs in all branches of mathematics.

Propositional Logic

Propositional Logic is the logic of compound statements built from simpler statements using so-called *Boolean connectives*.


George Boole (1815-1864)

Some applications in computer science:

- Design of digital electronic circuits.
- Expressing conditions in programs.
- Queries to databases & search engines.

Definition of a *Proposition*

Definition: A *proposition* (denoted *p*, *q*, *r*, …) is simply:

- a *statement* (*i.e.*, a declarative sentence)
  - *with some definite meaning*, (not vague or ambiguous)
- having a *truth value* that's either *true* (T) or *false* (F)
  - it is **never** both, neither, or somewhere "in between!"
    - However, you might not *know* the actual truth value,
    - and, the truth value might *depend* on the situation or context.
- In *probability theory*, we assign *degrees of certainty* ("between" T and F) to propositions.
  - But for now: think True/False only!

Examples of Propositions

- "It is raining." (In a given situation.)
- "Washington, D.C. is the capital of the U.S."
- "1 + 2 = 3"

But, the following are **NOT** propositions:

- "Who's there?" (interrogative, question)
- "La la la la la." (meaningless interjection)
- "Just do it!" (imperative, command)
- "Yeah, I sorta dunno, whatever..." (vague)
- "1 + 2" (expression with a non-true/false value)

Operators / Connectives

An *operator* or *connective* combines one or more *operand* expressions into a larger expression. (*E.g.*, “+” in numeric exprs.)

- *Unary* operators take 1 operand (*e.g.*, −3); *binary* operators take 2 operands (*e.g.*, 3 × 4).
- *Propositional* or *Boolean* operators operate on propositions (or their truth values) instead of on numbers.

Some Popular Boolean Operators

Formal Name	**Nickname**	**Arity**	**Symbol**
Negation operator	NOT	Unary	$\neg$
Conjunction operator	AND	Binary	$\wedge$
Disjunction operator	OR	Binary	$\vee$
Exclusive-OR operator	XOR	Binary	$\oplus$
Implication operator	IMPLIES	Binary	$\rightarrow$
Biconditional operator	IFF	Binary	$\leftrightarrow$

The Negation Operator

The unary *negation operator* "¬" (*NOT*) transforms a prop. into its logical *negation*.

E.g. If p = "I have brown hair."

then $\neg p$ = "I do **not** have brown hair."

The *truth table* for NOT:

p	$\neg p$
T	F
F	T

Operand column

Result column

T :≡ True; F :≡ False

":≡" means "is defined as"

The Conjunction Operator

The binary *conjunction operator* "$\wedge$" (*AND*) combines two propositions to form their logical *conjunction*.

$\wedge$ND

E.g. If p="I will have salad for lunch." and q="I will have steak for dinner.", then $p \wedge q$="I will have salad for lunch **and** I will have steak for dinner."

Remember: "$\wedge$" points up like an "A", and it means "$\wedge$ND"

Conjunction Truth Table

- Note that a conjunction $p_1 \wedge p_2 \wedge \ldots \wedge p_n$ of n propositions will have 2^n rows in its truth table.

Operand columns

p	q	$p \wedge q$
F	F	F
F	T	F
T	F	F
T	T	T

- Also: ¬ and ∧ operations together are suffi-cient to express *any* Boolean truth table!

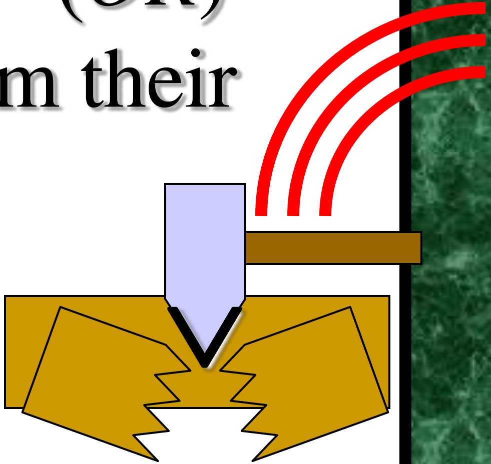
The Disjunction Operator

The binary *disjunction operator* "$\lor$" (*OR*) combines two propositions to form their logical *disjunction*.

p="My car has a bad engine."

q="My car has a bad carburetor."

$p \lor q$="Either my car has a bad engine, **or** my car has a bad carburetor."



Meaning is like "and/or" in English.

After the downward-pointing "axe" of "$\lor$" splits the wood, you can take 1 piece OR the other, or both.

Disjunction Truth Table

- Note that $p \lor q$ means that p is true, or q is true, **or both** are true!
- So, this operation is also called *inclusive or,* because it **includes** the possibility that both p and q are true.
- "¬" and "∨" together are also universal.

p	q	$p \lor q$
F	F	F
F	T	**T**
T	F	**T**
T	T	T

Note difference from AND

Nested Propositional Expressions

- Use parentheses to *group sub-expressions*: "I just saw my old *f*riend, and either he's grown or I've *s*hrunk." = $f \land (g \lor s)$
  - $(f \land g) \lor s$ would mean something different
  - $f \land g \lor s$ would be ambiguous
- By convention, "¬" takes *precedence* over both "∧" and "∨".
  - $\neg s \land f$ means $(\neg s) \land f$, **not** $\neg (s \land f)$

A Simple Exercise

Let p=“It rained last night”,
q=“The sprinklers came on last night,”
r=“The lawn was wet this morning.”

Translate each of the following into English:

$\neg p$ = “It didn’t rain last night.”

$r \wedge \neg p$ = “The lawn was wet this morning, and it didn’t rain last night.”

$\neg r \vee p \vee q$ = “Either the lawn wasn’t wet this morning, or it rained last night, or the sprinklers came on last night.”

The *Exclusive Or* Operator

The binary *exclusive-or operator* "$\oplus$" (*XOR*) combines two propositions to form their logical "exclusive or".

p = "I will earn an A in this course,"

q = "I will drop this course,"

$p \oplus q$ = "I will either earn an A in this course, or I will drop it (but not both!)"

Exclusive-Or Truth Table

- Note that $p \oplus q$ means that p is true, or q is true, but **not both**!
- This operation is called *exclusive or*, because it **excludes** the possibility that both p and q are true.
- "¬" and "⊕" together are **not** universal.

p	q	$p \oplus q$
F	F	F
F	T	T
T	F	T
T	T	**F**

Note difference from OR.

Natural Language is Ambiguous

Note that <u>English</u> "or" can be <u>ambiguous</u> regarding the "both" case!

"Pat is a singer or Pat is a writer." - $\vee$

"Pat is a man or Pat is a woman." - $\oplus$

p	q	p "or" q
F	F	F
F	T	T
T	F	T
T	T	?

Need context to disambiguate the meaning!

For this class, assume "or" means <u>inclusive</u>.

The *Implication* Operator

The *implication* $\overbrace{p}^{\text{antecedent}} \rightarrow \overbrace{q}^{\text{consequent}}$ states that p implies q.

I.e., If p is true, then q is true; but if p is not true, then q could be either true or false.

E.g., let p = "You study hard."
q = "You will get a good grade."

$p \rightarrow q$ = "If you study hard, then you will get a good grade." (else, it could go either way)

Implication Truth Table

- $p \to q$ is **false** only when p is true but q is **not** true.
- $p \to q$ does **not** say that p causes q!
- $p \to q$ does **not** require that p or q **are ever true**!
- *E.g.* "(1=0) → pigs can fly" is TRUE!

p	q	$p \to q$
F	F	T
F	T	T
T	F	**F**
T	T	T

The only False case!

Examples of Implications

- "If this lecture ever ends, then the sun will rise tomorrow." *[✓ True]* or *False*?
- "If Tuesday is a day of the week, then I am a penguin." *True* or *[✓ False]*?
- "If 1+1=6, then I am an ostrich." *[✓ True]* or *False*?
- "If the moon is made of green cheese, then I am richer than Bill Gates." *[✓ True]* or *False*?

English Phrases Meaning $p \rightarrow q$

- "*p* implies *q*"
- "if *p*, then *q*"
- "if *p*, *q*"
- "when *p*, *q*"
- "whenever *p*, *q*"
- "*q* if *p*"
- "*q* when *p*"
- "*q* whenever *p*"

- "*p* only if *q*"
- "*p* is sufficient for *q*"
- "*q* is necessary for *p*"
- "*q* follows from *p*"
- "*q* is implied by *p*"

We will see some equivalent logic expressions later.

Converse, Inverse, Contrapositive

Some terminology, for an implication $p \rightarrow q$:

- Its *converse* is: $q \rightarrow p$.
- Its *inverse* is: $\neg p \rightarrow \neg q$.
- Its *contrapositive*: $\neg q \rightarrow \neg p$.
- One of these three has the *same meaning* (same truth table) as $p \rightarrow q$. Can you figure out which?

PSN. Which one? Prove it.

The *biconditional* operator

The *biconditional* $p \leftrightarrow q$ states that p is true *if and only if (iff)* q is true.

p = "Joe Doe wins the 2020 senate election."

q = "Joe Doe will be senator for all of 2021."

$p \leftrightarrow q$ = "If, and only if, Joe Doe wins the 2020 election, Joe Doe will be senator for all of 2021."

Biconditional Truth Table

- $p \leftrightarrow q$ means that p and q have the **same** truth value.
- Note this truth table is the exact **opposite** of $\oplus$'s!

  Thus, $p \leftrightarrow q$ means $\neg(p \oplus q)$
- $p \leftrightarrow q$ does **not** imply that p and q are true, or that either of them causes the other, or that they have a common cause.

p	q	$p \leftrightarrow q$
F	F	T
F	T	F
T	F	F
T	T	T

Show that

a) $p \leftrightarrow q$ is the same as $p \rightarrow q$ and $q \rightarrow p$

b) Show that $p \leftrightarrow q$ is the same as $\neg(p \oplus q)$

Boolean Operations Summary

We have seen 1 unary operator and 5 binary operators. Their truth tables are below.

p	q	$\neg p$	$p\wedge q$	$p\vee q$	$p\oplus q$	$p\rightarrow q$	$p\leftrightarrow q$
F	F	T	F	F	F	T	T
F	T	T	F	T	T	T	F
T	F	F	F	T	T	F	F
T	T	F	T	T	F	T	T

TeЯЯible Joke

Professor - "In English, a double negative becomes a positive. But it is not true for every language. In Russian, a double negative still remains a negative. However, there is no language where a double positive can form a negative.“

Student - "yeah, right"

Logical Equivalence

Compound propositions (**formulas**) *p* and *q* are **logically equivalent** to each other, written $p \Leftrightarrow q$, iff *p* and *q* contain the same truth values as each other in **all** rows of their truth tables.

Proving Equivalence via Truth Tables

Ex. Prove that $p \lor q \Leftrightarrow \neg(\neg p \land \neg q)$.

p	q	$p \lor q$	$\neg p$	$\neg q$	$\neg p \land \neg q$	$\neg(\neg p \land \neg q)$
F	F	F	T	T	T	F
F	T	T	T	F	F	T
T	F	T	F	T	F	T
T	T	T	F	F	F	T

Equivalence Laws - Examples

- *Identity*: $p \wedge \mathbf{T} \Leftrightarrow p$ $p \vee \mathbf{F} \Leftrightarrow p$
- *Domination*: $p \vee \mathbf{T} \Leftrightarrow \mathbf{T}$ $p \wedge \mathbf{F} \Leftrightarrow \mathbf{F}$
- *Idempotent*: $p \vee p \Leftrightarrow p$ $p \wedge p \Leftrightarrow p$
- *Double negation:* $\neg\neg p \Leftrightarrow p$
- *Commutative:* $p \vee q \Leftrightarrow q \vee p$ $p \wedge q \Leftrightarrow q \wedge p$
- *Associative:* $(p \vee q) \vee r \Leftrightarrow p \vee (q \vee r)$
  $(p \wedge q) \wedge r \Leftrightarrow p \wedge (q \wedge r)$

More Equivalence Laws

- *Distributive*:

  $p\lor(q\land r) \Leftrightarrow (p\lor q)\land(p\lor r)$ (distribute $\lor$ over $\land$)

  $p\land(q\lor r) \Leftrightarrow (p\land q)\lor(p\land r)$ (distribute $\land$ over $\lor$)

- *De Morgan's*:

  $\neg(p\land q) \Leftrightarrow \neg p \lor \neg q$

  $\neg(p\lor q) \Leftrightarrow \neg p \land \neg q$

- *Trivial tautology/contradiction*:

  $p \lor \neg p \Leftrightarrow \mathbf{T}$ $p \land \neg p \Leftrightarrow \mathbf{F}$

Augustus De Morgan (1806-1871)

Defining Operators via Equivalences

Using equivalences, we can *define* operators in terms of other operators.

- Exclusive or: $p \oplus q \Leftrightarrow (p \vee q) \wedge \neg(p \wedge q)$
  $p \oplus q \Leftrightarrow (p \wedge \neg q) \vee (q \wedge \neg p)$
- Implies: $p \rightarrow q \Leftrightarrow \neg p \vee q$
- Biconditional: $p \leftrightarrow q \Leftrightarrow (p \rightarrow q) \wedge (q \rightarrow p)$
  $p \leftrightarrow q \Leftrightarrow \neg(p \oplus q)$

An Example Problem

- Check using a symbolic derivation whether $(p \wedge \neg q) \rightarrow (p \oplus r) \Leftrightarrow \neg p \vee q \vee \neg r$.

$(p \wedge \neg q) \rightarrow (p \oplus r)$ [Expand definition of $\rightarrow$]

$\Leftrightarrow \underline{\neg}(p \wedge \neg q)\ \underline{\vee}\ (p \oplus r)$ [Expand defn. of $\oplus$]

$\Leftrightarrow \neg(p \wedge \neg q) \vee (\underline{(p \vee r) \wedge \neg(p \wedge r)})$

[DeMorgan's Law]

$\Leftrightarrow \underline{(\neg p \vee q)} \vee ((p \vee r) \wedge \neg(p \wedge r))$

cont.

Example Continued...

$(\neg p \vee q) \vee ((p \vee r) \wedge \neg(p \wedge r))$ [$\vee$ commutes]

$\Leftrightarrow (q \vee \neg p) \vee ((p \vee r) \wedge \neg(p \wedge r))$ [$\vee$ associative]

$\Leftrightarrow q \vee (\neg p \vee ((p \vee r) \wedge \neg(p \wedge r)))$ [distrib. $\vee$ over $\wedge$]

$\Leftrightarrow q \vee (((\neg p \vee (p \vee r)) \wedge (\neg p \vee \neg(p \wedge r)))$

[assoc.] $\Leftrightarrow q \vee (((\neg p \vee p) \vee r) \wedge (\neg p \vee \neg(p \wedge r)))$

[trivial taut.] $\Leftrightarrow q \vee ((\mathbf{T} \vee r) \wedge (\neg p \vee \neg(p \wedge r)))$

[domination] $\Leftrightarrow q \vee (\mathbf{T} \wedge (\neg p \vee \neg(p \wedge r)))$

[identity] $\Leftrightarrow q \vee (\neg p \vee \neg(p \wedge r)) \Leftrightarrow$ *cont.*

End of Long Example

$q \lor (\neg p \lor \neg(p \land r))$

[DeMorgan's] $\Leftrightarrow q \lor (\neg p \lor (\neg p \lor \neg r))$

[Assoc.] $\Leftrightarrow q \lor ((\neg p \lor \neg p) \lor \neg r)$

[Idempotent] $\Leftrightarrow q \lor (\neg p \lor \neg r)$

[Assoc.] $\Leftrightarrow (q \lor \neg p) \lor \neg r$

[Commut.] $\Leftrightarrow \neg p \lor q \lor \neg r$

Q.E.D. (quod erat demonstrandum)

(Which was to be shown.)

PSN. Give alternate proof using truth tables that

$$(p \land \neg q) \to (p \oplus r) \Leftrightarrow \neg p \lor q \lor \neg r.$$

Review: Propositional Logic

- Atomic propositions: $p, q, r, \ldots$
- Boolean operators: $\neg \wedge \vee \oplus \rightarrow \leftrightarrow$
- Compound propositions: $(p \wedge \neg q) \vee r$
- Equivalences: $p \wedge \neg q \Leftrightarrow \neg(p \rightarrow q)$
- Proving equivalences using:
  - Truth tables.
  - Symbolic derivations. $p \Leftrightarrow q \Leftrightarrow r \ldots$

Tautologies and Contradictions

A *tautology* is a compound proposition that is **true** *no matter what* the truth values of its atomic propositions are!

Ex. $p \lor \neg p$

A *contradiction* is a compound proposition that is **false** no matter what! *Ex.* $p \land \neg p$

Other compound props. are *contingencies*.

Interpretation

Definition. An **interpretation** is an assignment I of a truth value, i.e, T or F, to every propositional letter *r*. We denote the assignment of a truth value to *r* by *I*(*r*).

Tautologies and Contradictions

φ is a tautology iff $I(\varphi) = \mathrm{T}$ for every interpretation I.

φ is a contradiction iff $I(\varphi) = \mathrm{F}$ for every interpretation I.

φ is a tautology iff $\neg\varphi$ is a contradiction.

$p \Leftrightarrow q$ iff the compound proposition $p \leftrightarrow q$ is a tautology.

Show the following compound formula is a contradiction:

$(p \vee q \vee r) \wedge (p \vee q \vee \neg r) \wedge (p \vee \neg q \vee r) \wedge (\neg p \vee q \vee r) \wedge$

$(p \vee \neg q \vee \neg r) \wedge (\neg p \vee q \vee \neg r) \wedge (\neg p \vee \neg q \vee r) \wedge (\neg p \vee \neg q \vee \neg r)$

Proof. $(p \wedge \neg p) \vee (q \wedge \neg q) \vee (r \wedge \neg r) = \mathrm{F} \vee \mathrm{F} \vee \mathrm{F} = \mathrm{F}$

Using the distributive law of $\vee$ over $\wedge$ we have

$\mathrm{F} = (p \wedge \neg p) \vee (q \wedge \neg q) \vee (r \wedge \neg r)$

$= (p \vee q \vee r) \wedge (p \vee q \vee \neg r) \wedge (p \vee \neg q \vee r) \wedge (\neg p \vee q \vee r) \wedge$

$(p \vee \neg q \vee \neg r) \wedge (\neg p \vee q \vee \neg r) \wedge (\neg p \vee \neg q \vee r) \wedge (\neg p \vee \neg q \vee \neg r)$

PSN. Show the following compound formula is a tautology:

$= (p \wedge q \wedge r) \vee (p \wedge q \wedge \neg r) \vee (p \wedge \neg q \wedge r) \vee (\neg p \wedge q \wedge r) \vee$

$(p \wedge \neg q \wedge \neg r) \vee (\neg p \wedge q \wedge \neg r) \vee (\neg p \wedge \neg q \wedge r) \vee (\neg p \wedge \neg q \wedge \neg r)$

Satisfiability

A formula φ is **satisfiable** iff it is T for some interpretation *I*, i.e., *I*(φ) = T. Otherwise it is **unsatisfiable**.

Every tautology φ is satisfiable.

φ is **not satisfiable** iff φ is a contradiction.

GENIE: "Because you freed me from the lamp, I grant unto you one wish."

ME: "Can I wish for anything?"
GENIE: "Yes, anything."
ME: "Literally anything?"
GENIE: "Literally anything."
ME: "And you'll do it?"
GENIE: "I'm a genie, it's what I do."
ME (after some thought): "I wish for this wish to not be granted."
GENIE: "But wait! I can only grant your wish by not granting it, but by not granting it I'm actually granting it, but that means I have to not grant it, so...LOGIC ERROR...DOES NOT COMPUTE..." (explodes)

Representations of Formulas

Chapter 2 of Textbook

- Normal Forms, Section 2.5 pp. 121-127
  - Disjunctive Normal Form (DNF)
  - Conjunctive Normal Form (CNF)
- Expression Trees, page 94 of text
- Combinatorial Circuits, using gates to represent formulas, page 98

Normal Forms

Formula *f*

Disjunctive Normal Form (DNF), pp. 122-125

Formula *f* is expression as a **disjunction** of clauses, where each clause is a conjunction of positive and negative literals.

Conjunctive Normal Form (CNF), pp. 125-127

Formula *f* is expressed as a **conjunction** of clauses, where each clause is a disjunction of positive and negative literals.

Disjunctive Normal Form (DNF)

p q r	f	Clause Conjunction
F F F	**T**	$\neg p \wedge \neg q \wedge \neg r$
F F T	F	$\neg p \wedge \neg q \wedge r$
F T F	**T**	$\neg p \wedge q \wedge \neg r$
F T T	**T**	$\neg p \wedge q \wedge r$
T F F	F	$p \wedge \neg q \wedge \neg r$
T F T	F	$p \wedge \neg q \wedge r$
T T F	**T**	$p \wedge q \wedge \neg r$
T T T	**T**	$p \wedge q \wedge r$

$$f \Leftrightarrow (\neg p \wedge \neg q \wedge \neg r) \vee (\neg p \wedge q \wedge \neg r) \vee (\neg p \wedge q \wedge r) \vee (p \wedge q \wedge \neg r) \vee (p \wedge q \wedge r)$$

Conjunctive Normal Form (CNF)

$p \quad q \quad r$	f	$\neg f$	Clause Conjunction
F F F	T	F	$\neg p \land \neg q \land \neg r$
F F T	F	**T**	$\neg p \land \neg q \land r$
F T F	T	F	$\neg p \land q \land \neg r$
F T T	T	F	$\neg p \land q \land r$
T F F	F	**T**	$p \land \neg q \land \neg r$
T F T	F	**T**	$p \land \neg q \land r$
T T F	T	F	$p \land q \land \neg r$
T T T	T	F	$p \land q \land r$

First put negation of formula in DNF:

$$\neg f \Leftrightarrow (\neg p \land \neg q \land r) \lor (p \land \neg q \land \neg r) \lor (p \land \neg q \land r)$$

Conjunctive Normal Form (CNF) cont'd

$\neg f \Leftrightarrow (\neg p \wedge \neg q \wedge r) \vee (p \wedge \neg q \wedge \neg r) \vee (p \wedge \neg q \wedge r)$

$f \Leftrightarrow \neg(\neg f)$

$\Leftrightarrow \neg((\neg p \wedge \neg q \wedge r) \vee (p \wedge \neg q \wedge \neg r) \vee (p \wedge \neg q \wedge r))$

(apply De Morgan's Law for $\vee$)

$\Leftrightarrow \neg(\neg p \wedge \neg q \wedge r) \wedge \neg(p \wedge \neg q \wedge \neg r) \wedge \neg(p \wedge \neg q \wedge r))$

(apply De Morgan's Law for $\wedge$ to each clause)

$\Leftrightarrow (\neg\neg p \vee \neg\neg q \vee \neg r) \wedge (\neg p \vee \neg\neg q \vee \neg\neg r) \wedge (\neg p \vee \neg\neg q \vee \neg r))$

$\Leftrightarrow (p \vee q \vee \neg r) \wedge (\neg p \vee q \vee r) \wedge (\neg p \vee q \vee \neg r)$

Formula *f* is now expressed in CNF as a conjunction of clauses, where each clause is a disjunction of positive and negative literals.

Put following formula f in DNF:

$$(p \wedge \neg q) \leftrightarrow (p \oplus r)$$

Put same formula *f* in CNF:

$$(p \land \neg q) \leftrightarrow (p \oplus r)$$

CNF SAT and NP-complete

CNF SAT is the quintessential NP-complete problem.

P = NP? Most important problem in computer science and mathematics.

We will discuss in the next lecture.

Other representations of Formulas

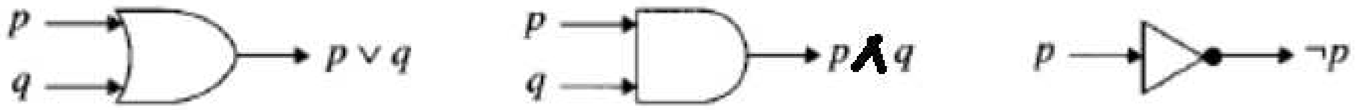
- **Expression Trees**, page 94 of text
- **Combinatorial Circuits**, using gates to represent formulas, page 98

Expression Trees

PSN. Give the expression tree for the formula

$$(((p \vee q) \wedge (\neg r)) \vee (q \vee r))$$

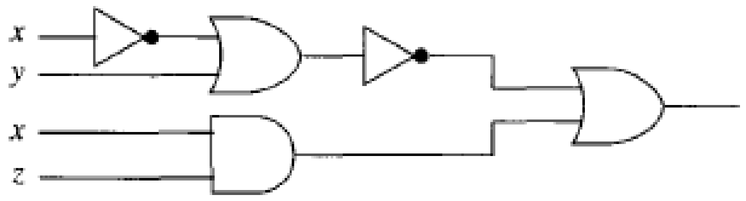
Gates to represent formulas



OR, AND and NOT Gates

Combinatorial Circuit

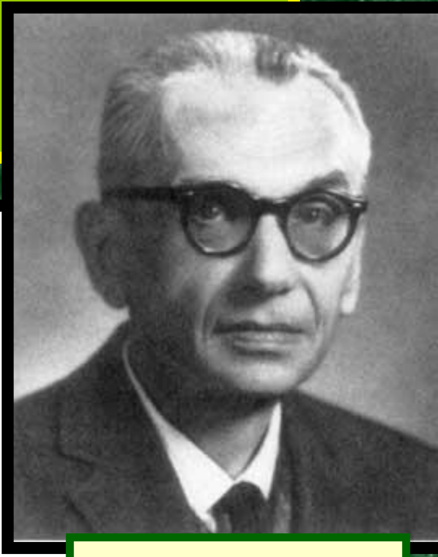
A set of gates is called a **combinatorial circuit** or **combinatorial network.**



PSN. Obtain formula associated with the above combinatorial circuit.

Predicate Logic

- *Predicate logic* is an extension of propositional logic that permits concisely reasoning about whole *classes* of entities.
- Propositional logic (recall) treats simple *propositions* (sentences) as atomic entities.
- In contrast, *predicate* logic distinguishes the *subject* of a sentence from its *predicate*.
  - Remember these English grammar terms?


Kurt Gödel
1906-1978

- Predicate logic is the foundation of the field of *mathematical logic*, which culminated in *Gödel's incompleteness theorem*, which revealed the ultimate limits of mathematical thought:
  - Given any finitely describable, consistent proof procedure, there will always remain *some* true statements that will *never be proven* by that procedure.
- *i.e.*, we can't discover *all* mathematical truths, unless we sometimes resort to making *guesses*.

Subjects and Predicates

- In the sentence "The dog is sleeping":
  - The phrase "the dog" denotes the *subject* - the *object* or *entity* that the sentence is about.
  - The phrase "is sleeping" denotes the *predicate*- a property that is true **of** the subject.
- In predicate logic, a *predicate* is modeled as a *function* $P(\cdot)$ from objects to propositions.
  - $P(x)$ = "x is sleeping" (where x is any object).

More About Predicates

- Convention: Lowercase variables *x*, *y*, *z*... denote objects/entities; uppercase variables *P*, *Q*, *R*… denote propositional functions (predicates).
- Keep in mind that the *result of applying* a predicate *P* to an object *x* is the *proposition* *P*(*x*). But the predicate *P* **itself** (*e.g.* *P* = "is sleeping") is **not** a proposition (not a complete sentence).
  - *E.g.* if *P*(*x*) = "*x* is a prime number", *P*(3) is the *proposition* "3 is a prime number."

Universes of Discourse (U.D.s)

- The power of distinguishing objects from predicates is that it lets you state things about *many* objects at once.
- E.g., let $P(x)$="$x+1>x$". We can then say, "For *any* number x, $P(x)$ is true" instead of $(0+1>0) \wedge (1+1>1) \wedge (2+1>2) \wedge \ldots$
- The collection of values that a variable x can take is called x's *universe of discourse*.

Quantifier Expressions

- *Quantifiers* provide a notation that allows us to *quantify* (count) *how many* objects in the univ. of disc. satisfy a given predicate.
- "$\forall$" is the FOR$\forall$LL or *universal* quantifier. $\forall x\, P(x)$ means *for all* x in the u.d., P holds.
- "$\exists$" is the $\exists$XISTS or *existential* quantifier. $\exists x\, P(x)$ means there *exists* an x in the u.d. (that is, 1 or more) such that $P(x)$ is true.

The Universal Quantifier ∀

- Example:
  Let the u.d. of x be parking spaces at UC.
  Let $P(x)$ be the *predicate* "x is full."
  Then the *universal quantification of* $P(x)$, $\forall x\, P(x)$, is the *proposition:*
  - "All parking spaces at UC are full."
  - *i.e.*, "Every parking space at UC is full."
  - *i.e.*, "For each parking space at UC, that space is full."

The Existential Quantifier ∃

- Example:
  Let the u.d. of *x* be parking spaces at UC.
  Let *P*(*x*) be the *predicate* "*x* is full."
  Then the *existential quantification of P*(*x*), ∃*x P*(*x*), is the *proposition*:
  - "Some parking space at UC is full."
  - "There is a parking space at UC that is full."
  - "At least one parking space at UC is full."

Calculus Example

- One way of precisely defining the calculus concept of a *limit*, using quantifiers:

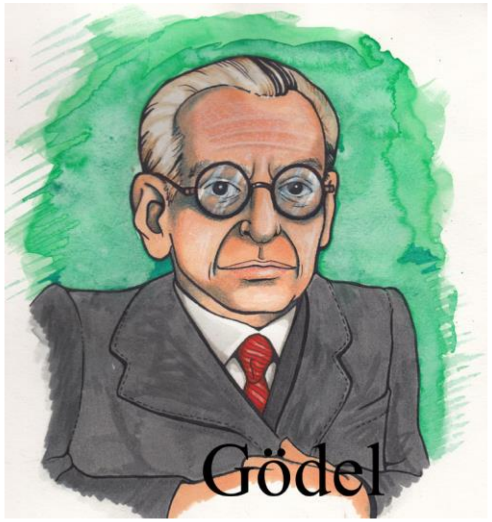
$$\left(\lim_{x\to a} f(x) = L\right) \Leftrightarrow$$

$$\left(\begin{array}{l}\forall \varepsilon > 0 : \exists \delta > 0 : \forall x : \\ \left(|x-a| < \delta\right) \to \left(|f(x)-L| < \varepsilon\right)\end{array}\right)$$

Good Ole Days

No one uses logic anymore.

I miss the Godel days.



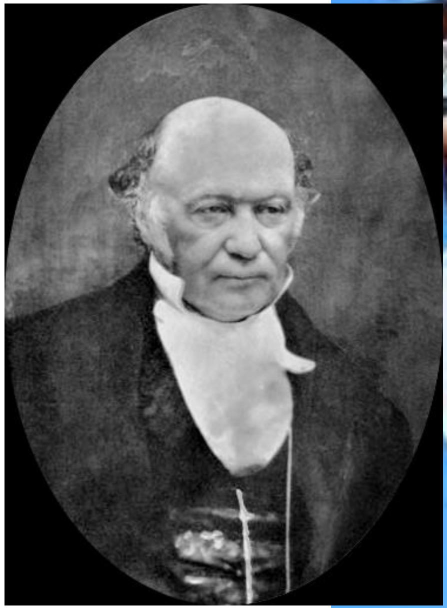
NP-Completeness

Textbook reading:

Section 2.5.6, Page 129

Hamiltonian Cycle

Sir William Rowen Hamilton's Icosian Game



Goal of Icosian Game

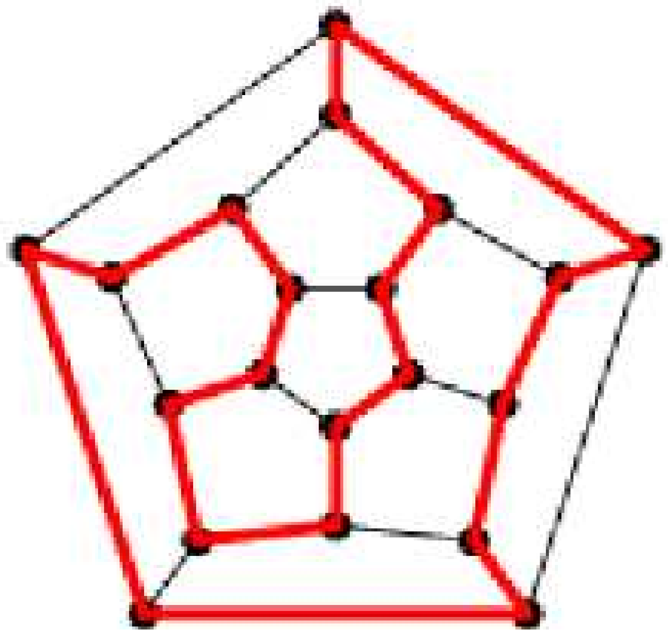
Vertices of the icosahedron represent cities. The goal is to perform a tour of the 20 cities and return to the starting vertex, following an edge of the icosahedron to move between two cities.

This is done by placing pegs on the board so that Peg i and Peg $i + 1$, $i = 1, 2, \ldots, 19$, and Peg 20 and Peg 1 are on adjacent positions, i.e., end vertices of an edge of the icosahedron.

Can you solve the problem?

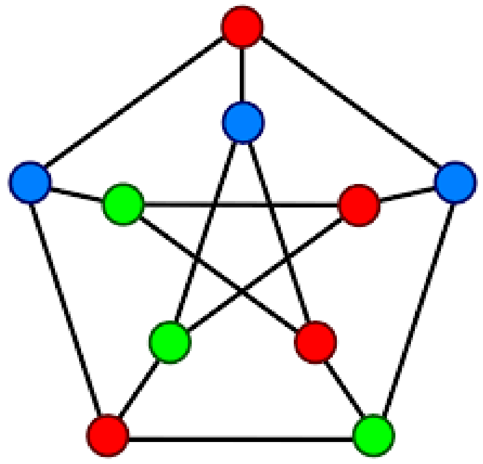
Solution to Icosian Game

Solution involves finding a **Hamiltonian cycle** in the icosahedron:



Coloring Problem

Given *k* colors, color the vertices of a graph so that every edge is **properly** colored, i.e., both ends of the edge are colored differently. Below is a properly vertex 3-colored graph.



Clique Problem

Consider the Friendship Network on Facebook, i.e., the vertex set *V* are users of Facebook and the edges set *E* consists of all pairs of people that are friends. A **clique** is a group of people, such that every pair are friends, i.e., form a clique. In graph theory, a ***k*-clique** is a subset of *k* vertices that that every pair is joined with an edge.

Given a positive integer *k*, does there exists a *k*-clique, i.e., *k* people, such that every pair are friends.

Sum of Subsets

Given the input integers $A = \{a_0, a_1, . . . , a_{n\text{-}1}\}$, and the target sum c, is there are subset of the integers whose sum equals c?

No Worst-Case Polynomial-Time Algorithms Known

Mathematicians worked for years on trying to obtain efficient algorithms for solving these problems in general and many other natural problems without success. In fact, to this day, no known worst-case polynomial algorithms are known for solving any of them. It turns out that they are all **NP-hard**.

Decision version of the problem

The decision version of a problem asks whether a solution to the problem exists, “yes” or “true” if its exists, “no” or “false”, otherwise. For example,

- Does graph *G* contain a Hamiltonian cycle?
- Can graph *G* be properly vertex 3-colored?
- Does graph *G* contain a *k*-clique?
- Does there exists a subset of a set *A* whose elements sum to *c*?

Class P

A decision problem is in *P* if it can be solved with a polynomial-time algorithm, i.e., an algorithm that for any input of size *n* can be solved in polynomial time, i.e., in time at most n^k for some positive integer *k*.

These problems are sometimes called ***tractable***. The class P is the set of all such problems.

Class NP

NP stands for **nondeterministic polynomial.** It applies to decision problems.

Given a decision problem, we associate a **certificate** with the problem. Examples:

Hamiltonian Cycle Problem. Certificate is sequence of *n* distinct vertices: “yes-certificate” if it is corresponds to a Hamiltonian cycle; otherwise “no-certificate”.

3-Coloring Problem. Certificate is 3-coloring of vertices: “yes-certificate” if it is a proper coloring; otherwise, it is a “no-certificate”.

Clique Problem. Certificate is subset of *k* vertices: “yes-certificate” if it is a *k*-clique; otherwise it is a “no-certificate”.

Sum of Subsets Problem. Certificate is a subset S of A: “yes-certificate” if the elements of A sum to *c*; otherwise it is a “no-certificate”.

High-Level Pseudocode for NP

```
function NPAlgorithm(A,I)
Input:  A (a decision problem), I (an instance of problem A)
Output:  “yes” or “don’t know”
   1.  In polynomial time, guess a candidate certificate C for the
                                                       problem A
   2.   In polynomial time, use C to deterministically verify that I is
                                                       a yes instance.
     if a yes instance is verified in step 2 then
           return (“yes”)
     else
           return(“don’t know”)
     endif
end NPAlgorithm
```

NP algorithms

Hamiltonian Cycle Problem. Certificate is sequence of *n* distinct vertices $u_1, u_2, \ldots, u_n$ in the graph *G*. It can be verified in time *n* whether this certificate is a "yes-certificate", i.e., is a Hamiltonian cycle by simply checking that $\{u_i, u_{i+1}\}$, $i = 1, 2, \ldots, n-1$, and $\{u_n, u_1\}$, are all edges of *G.*

Coloring Problem. It can be verified whether a 3-coloring is proper in time *m*, where *m* is the number of edges of *G*, by scanning all the edges to see whether both end vertices of the edge are colored the same.

Clique Problem. It can be verified whether a subset of *k* vertices forms a *k*-clique in time $k(k-1)/2$ by checking whether every pair of vertices of the subset is adjacent in *G*.

Sum of Subsets Problem. It can be verified whether the elements of a subset *S* of a set *A* of size *n* sum to *c* by performing $|S| - 1 < n$ additions.

For each of these problems a candidate certificate that is guessed can be verified to be a yes-certificate or no-certificate in polynomial time, so **they are all in NP**.

Polynomial-Time Reducibility

Given two decision problems *A* and *B*, we say that *A* is (polynomially) *reducible* to *B*, denoted $A \propto B$, if there is a mapping *f* from the inputs to problem *A* to the inputs to problem *B*, such that

1. The function *f* can be computed in polynomial time (that is, there is a polynomial algorithm that for input I to problem A outputs the input *f*(*I*) to problem *B*), and

2. the answer to a given input I to problem *A* is yes if, and only if, the answer to the input *f*(*I*) to problem B is yes.

Properties of Reduction Relation $\propto$

- The relation $\propto$ is transitive; that is, if $A \propto B$ and $B \propto C$ then $A \propto C$.

- If $A \propto B$ and B has polynomial complexity, then so does A.

Crazy Definition?

A problem *B* is **NP-complete**, if it is in NP and **any** problem *A* in NP is reducible to *B*, i.e., $A \propto B$.

At first, this definition may seem crazy, because it suggests that you can use *B* to solve every other problem in NP? It would mean that if *B* is in P, every NP problem is in P, i.e., P = NP.

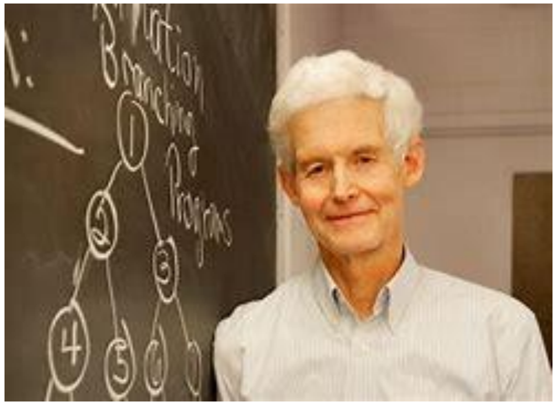
NP-Complete Problems Exist!

Surprising it was shown by Stephen Cook and independently by Leonid Levin, that NP-complete problems exist!

Most Famous Theorem in all of Mathematics and Computer Science

Cook–Levin Theorem. CNF SAT is NP-complete.

Discovered independently by Stephen Cook and Lenoid Levin in 1971.



The Reduction CNF SAT $\propto$ CNF 3-SAT.

Consider any instance of CNF SAT

$$I = C_1 \wedge C_2 \wedge . . . \wedge C_m.$$

C_i is of the form $y_1 \vee \cdots \vee y_j$ where

$y_1, \dots, y_j \in \{ x_1, \dots, x_n\} \cup \{\bar{x}_1, \dots, \bar{x}_n\}$ and $\bar{x}_i = \neg x_i$

We need to construction a reduction *f* mapping an input *I* of CNF SAT to an input *I'* of CNF 3-SAT, so that *I* is satisfiable iff *I'* is satisfiable.

PSN.

a) for a clause C of size 1, i.e., C is x (where x is a positive or negative literal), find a conjunction of clauses of size 3 that is logically equivalent using new variables y and z.

b) for a clause C of size 2, i.e., C is $x \vee y$ (where x and y are positive or negative literals), find a conjunction of clauses of size 3 that is logically equivalent using new variable z.

Reduction from CNF SAT to CNF 3-SAT

We construct instance *I'* of CNF 3-SAT from instance *I* of SAT by replacing clauses of size 1 and 2 using the logically equivalent conjunction of clauses from PSN problem we just did. Otherwise, we replace $C_i = y_1 \vee \cdots \vee y_j$ with

f = $(y_1 \vee y_2 \vee z_1) \wedge (\bar{z}_1 \vee y_3 \vee z_2) \wedge (\bar{z}_2 \vee y_4 \vee z_3) \wedge (\bar{z}_2 \vee y_5 \vee z_3) \wedge \cdots \wedge (\bar{z}_{j-2} \vee y_{j-1} \vee y_j)$

If *I* is satisfiable then at least one of $y_1, \dots, y_j$ are true, and the z_i's can be chosen so the formula *f* is true. On the other hand if *I* is not satisfiable, then for any truth assignment some clause C_i is false, i.e., the literals $y_1, \dots, y_j$ are all false. In which case no matter what values are chosen for the variables z_i, *f* has the value false, so that *I'* is not satisfiable (convince yourself of this).

NP-complete

Hundreds of natural and important problems have been shown to be NP-complete, including Vertex Coloring and Sum of Subsets. In fact, the coloring problem is NP-complete even for 3 colors.

NP-Hard.

A problem, not necessary a decision problem, is ***NP-hard***, if it could be applied to solve an NP-complete problem in polynomial time, i.e., it is at least as difficult as an NP-complete problem.

There are thousands of important and practical problems that are known to be NP-hard. In fact, most algorithmic problems that are encountered in practice in science and engineering are NP-hard.

P = NP?

- This is one of the most celebrated and important problems in all of computer science and mathematics.
- To show P = NP, all you would have to do is design a polynomial-time algorithm for **one** NP-complete problem.
- Some of the greatest minds in mathematics and computer science have tried to crack this problem for about 50 years now, without success.
- The conjecture these days is that P ≠ NP.

Bagel Challenge

If you solve the problem of whether P = NP, you will get an A in the course.

Further, I will buy you a baker’s dozen bagels

and write you are check for

$$\$e^{i\pi}+1$$

Functions

Textbook Reading

Chapter 4, pp. 219-235

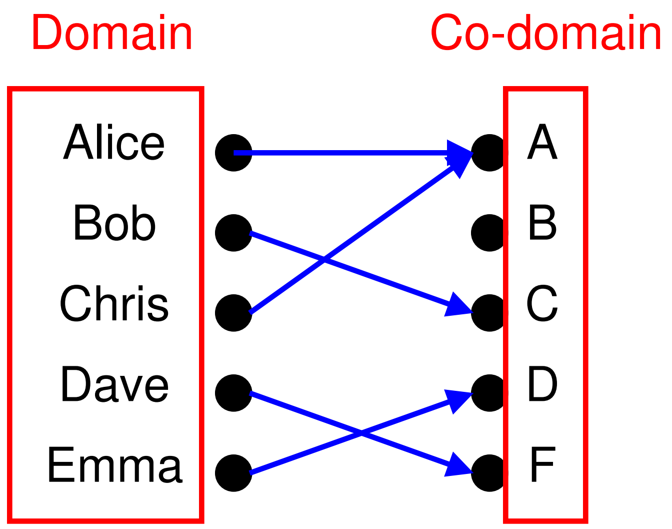
Section 4.6, pp. 253-257 (Pigeonhole Principle)

Definition of a function

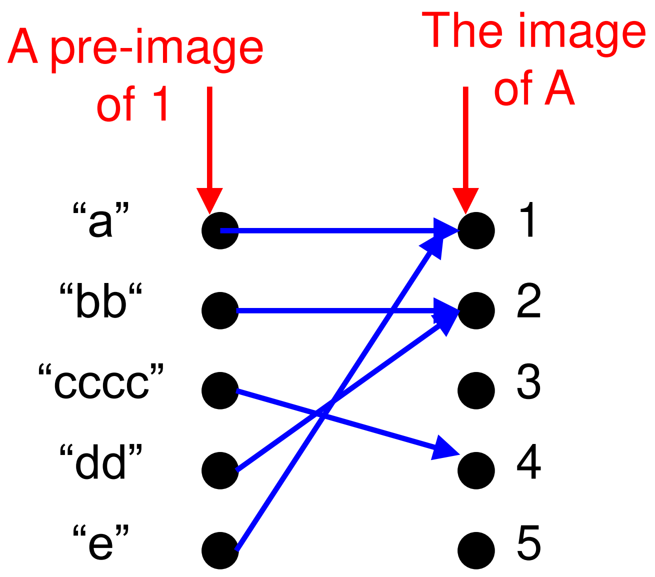
- A function is a binary relation between two sets A (domain) and B (co-domain), i.e., a subset of $A \times B$, so that each element in A occurs in exactly one pair.

- A function takes an element from a set and maps it to a UNIQUE element in another set.

Examples of functions

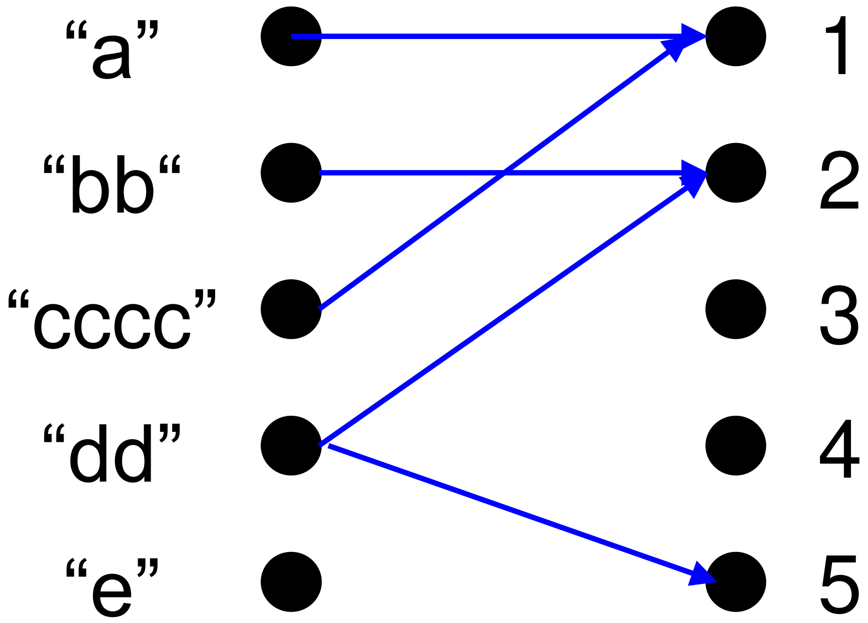


A class grade function

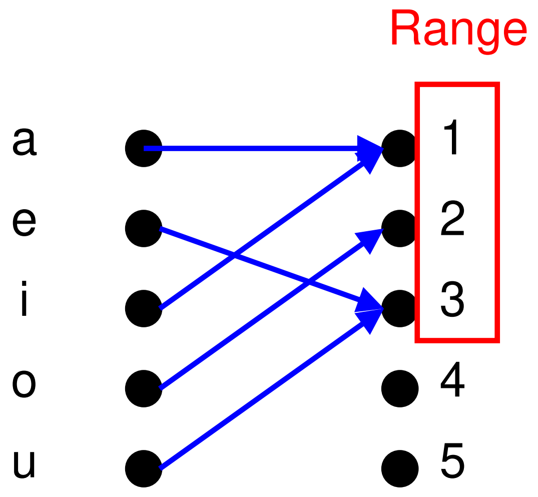


A string length function

Not a function



Range



Some function…

Function arithmetic

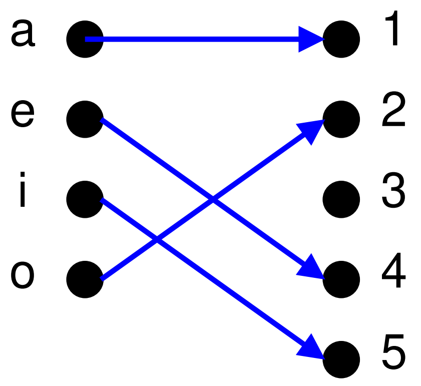
- Let $f_1(x) = 2x$
- Let $f_2(x) = x^2$

- $f_1+f_2 = (f_1+f_2)(x) = f_1(x)+f_2(x) = 2x+x^2$

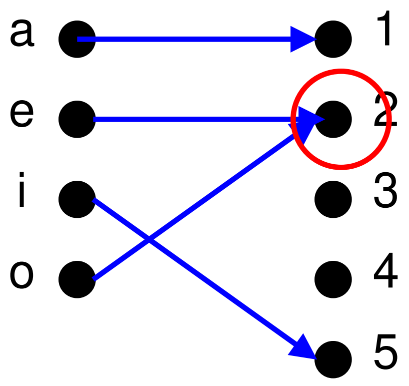
- $f_1*f_2 = (f_1*f_2)(x) = f_1(x)*f_2(x) = 2x*x^2 = 2x^3$

One-to-one functions

- A function is one-to-one if each element in the range has a unique pre-image



A one-to-one function

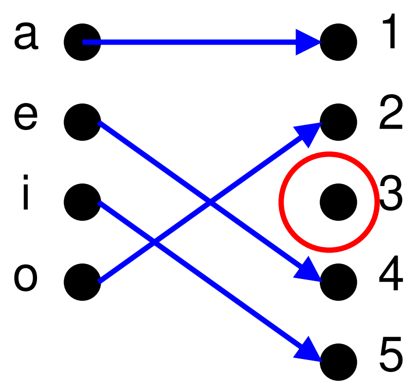


A function that is not one-to-one

More on one-to-one

- Injective is synonymous with one-to-one
  - "A function is **injective**"
- A function is an **injection** if it is one-to-one

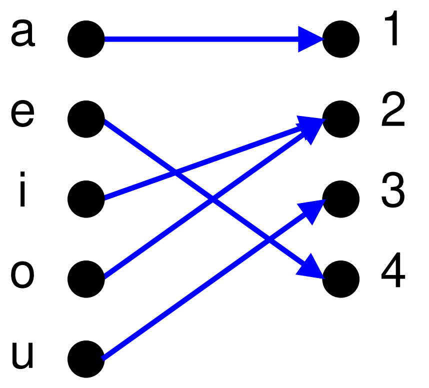
- Note that there can be un-used elements in the co-domain



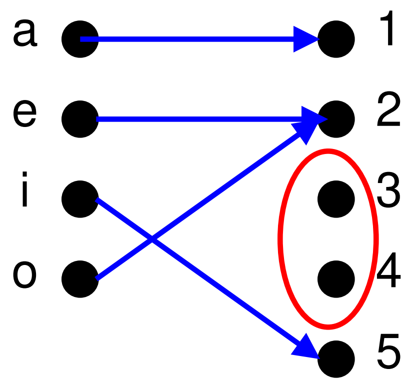
A one-to-one function

Onto functions

- A function is **onto** if each element in the co-domain is an image of some pre-image



An onto function

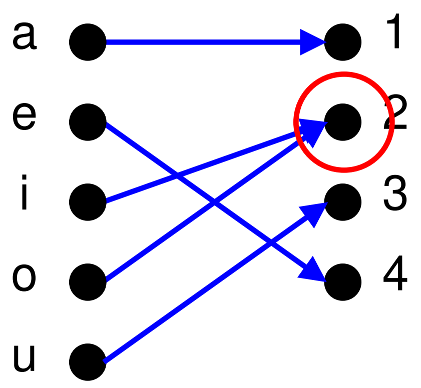


A function that is not onto

More on onto

- Surjective is synonymous with onto
  - "A function is **surjective**"
- A function is a **surjection** if it is onto

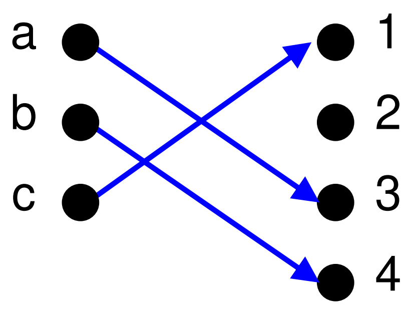
- Note that there can be multiply used elements in the co-domain



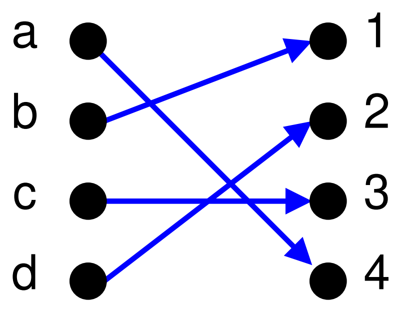
An onto function

Onto vs. one-to-one

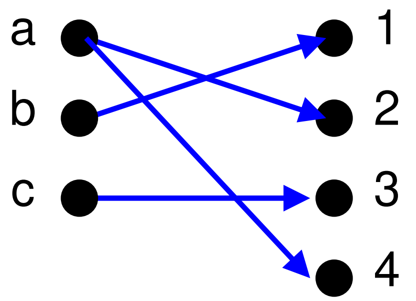
- Are the following functions onto, one-to-one, both, or neither?



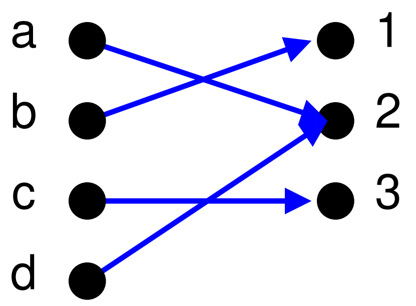
1-to-1, not onto



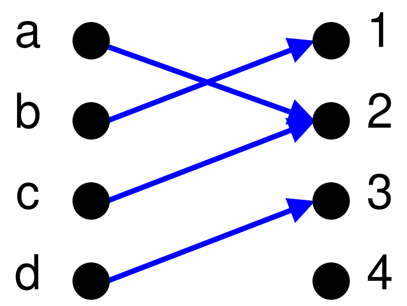
Both 1-to-1 and onto



Not a valid function



Onto, not 1-to-1

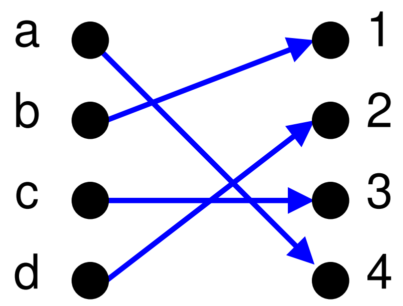


Neither 1-to-1 nor onto

Bijections

- Consider a function that is both one-to-one and onto:

- Such a function is a **bijection**



Identity functions

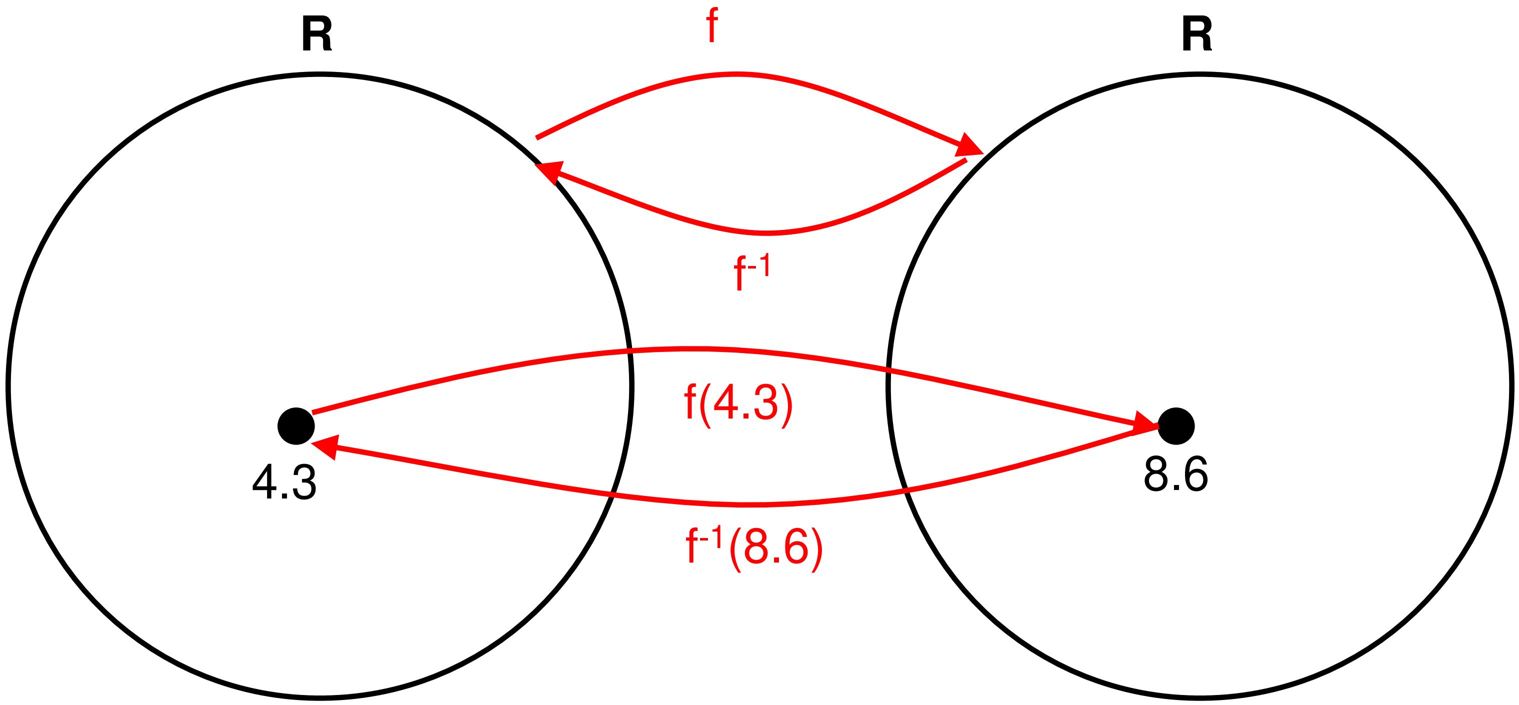
- A function such that the image and the pre-image are ALWAYS equal

- $f(x) = 1*x$
- $f(x) = x + 0$

- The domain and the co-domain must be the same set

Inverse functions

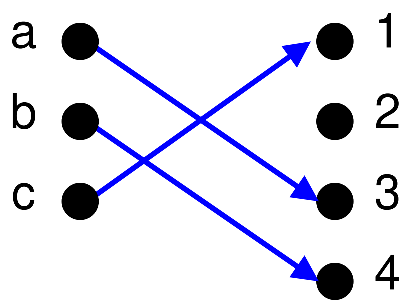
Let f(x) = 2*x



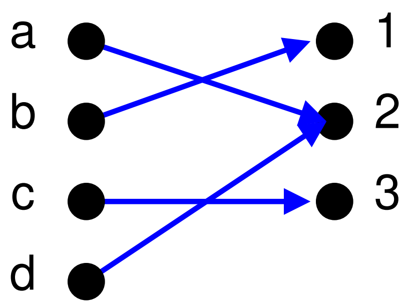
Then $f^{-1}(x) = x/2$

More on inverse functions

- Can we define the inverse of the following functions?



What is $f^{-1}(2)$?
Not onto!



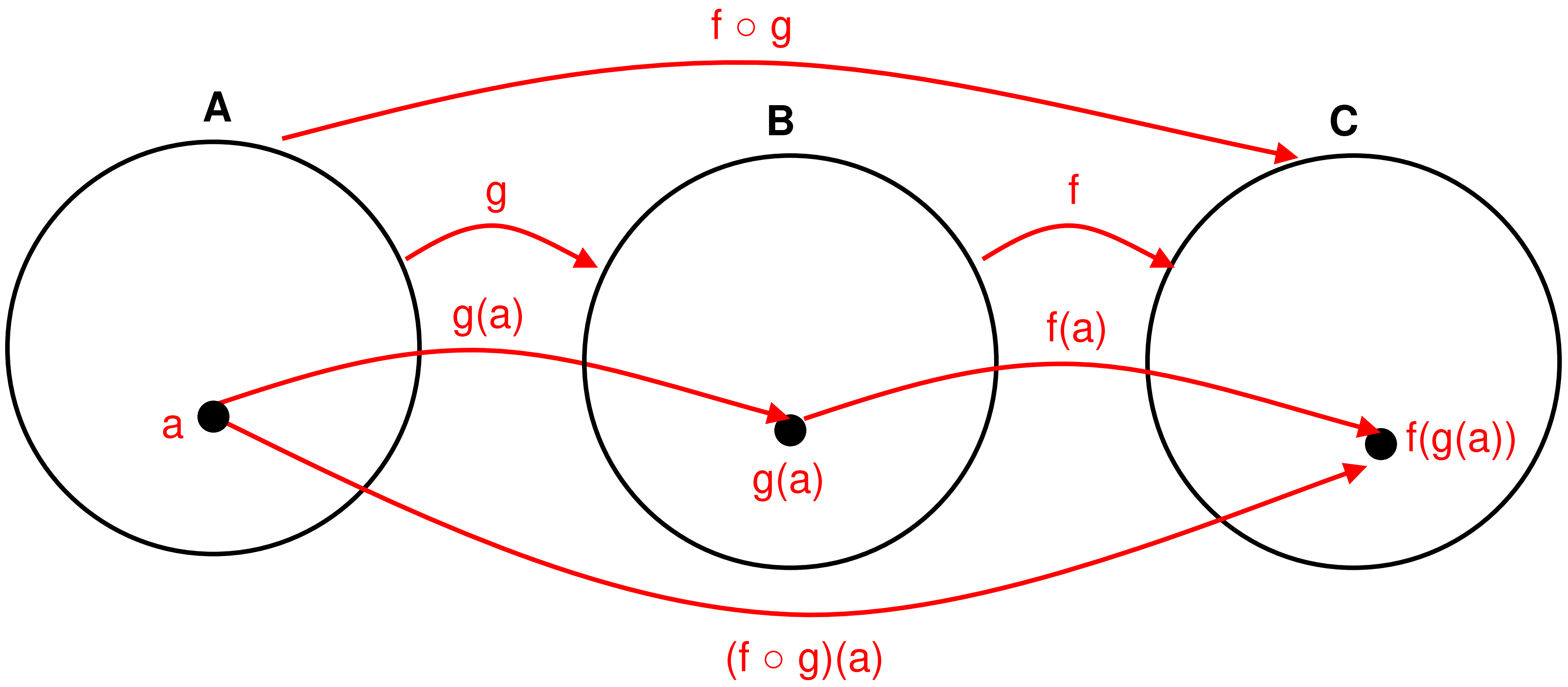
What is $f^{-1}(2)$?
Not 1-to-1!

- An inverse function is ONLY defined on a bijection

Compositions of functions

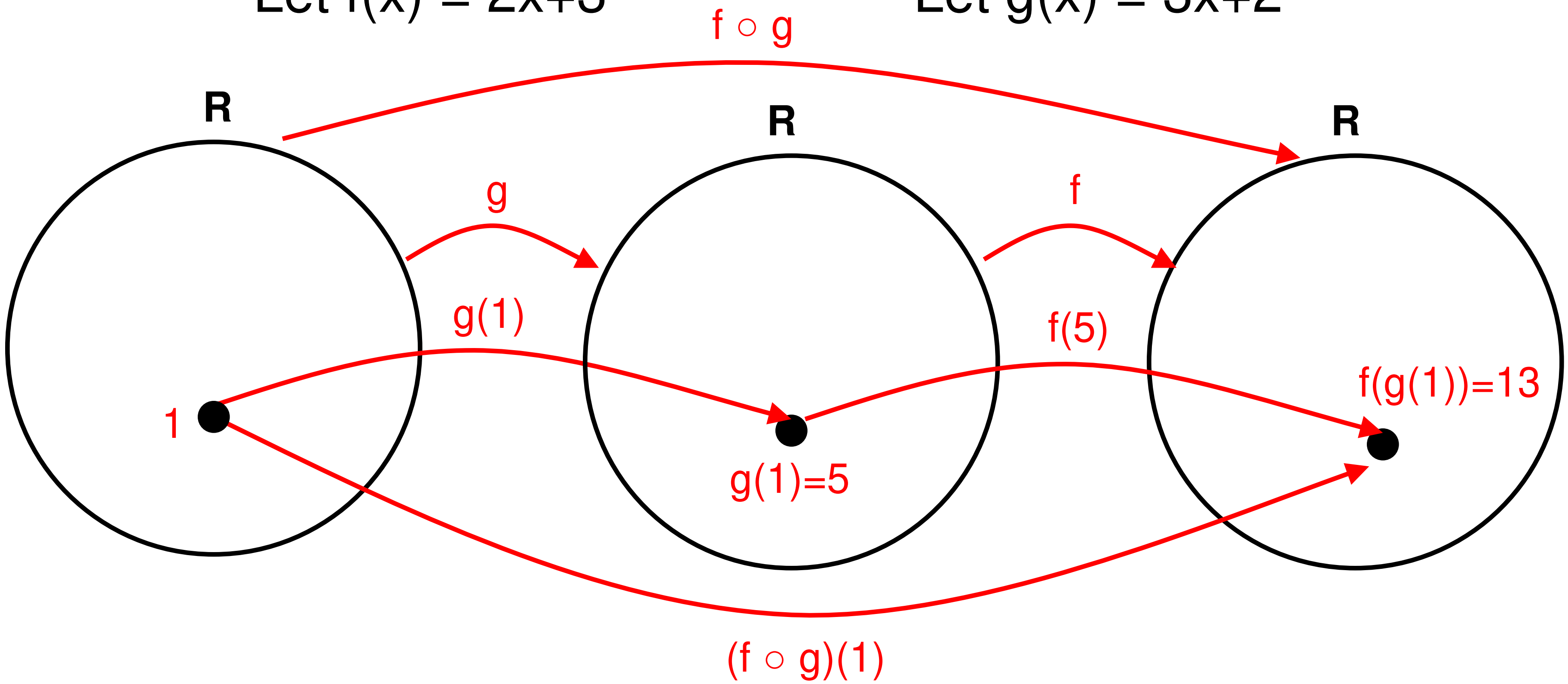
- Let $(f \circ g)(x) = f(g(x))$
- Let $f(x) = 2x+3$ Let $g(x) = 3x+2$
- $g(1) = 5$, $f(5) = 13$
- Thus, $(f \circ g)(1) = f(g(1)) = 13$

Compositions of functions



Compositions of functions

Let $f(x) = 2x+3$ Let $g(x) = 3x+2$



$f(g(x)) = 2(3x+2)+3 = 6x+7$

Compositions of functions

Does f(g(x)) = g(f(x))?

Let f(x) = 2x+3 Let g(x) = 3x+2

f(g(x)) = 2(3x+2)+3 = 6x+7

g(f(x)) = 3(2x+3)+2 = 6x+11

Not equal!

Function composition is not commutative!

PSN: Matrix Multiplication

Consider the functions (linear transformations)

$$A(x_1, x_2) = \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} a_{00}x_0 + a_{01}x_1 \\ a_{10}x_0 + a_{11}x_1 \end{bmatrix}$$

$$B(x_1, x_2) = \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} b_{00}x_0 + b_{01}x_1 \\ b_{10}x_0 + b_{11}x_1 \end{bmatrix}$$

Compute $A \circ B$. Discuss.

Matrix Multiplication corresponds to composition of linear function

The matrix product of a $p \times q$ matrix A and a $q \times r$ matrix B corresponds to the composition of a linear transformation A from R^p to R^q and a linear transformation B from R^q to R^r where R^p, R^q, R^r denote the set of all vectors of dimensions p, q and r, respectively.

Inverse of Composition

PSN. Let f be an invertible function from Y to Z and g be an invertible function from X to Y.

Show that the inverse of $f \circ g$ is:

$$(f \circ g)^{-1} = g^{-1} \circ f^{-1}$$

Floor, Ceiling and Round Functions

- Floor: $\lfloor x \rfloor$ means take the greatest integer less than or equal to the number
- Ceiling: $\lceil x \rceil$ means take the lowest integer greater than or equal to the number
- round(x) = floor(x + 0.5)

The Pigeonhole Principle or Dirichlet's drawer principle

- Suppose a flock of pigeons fly into a set of pigeonholes to roost.
- If there are more pigeons than pigeonholes, then there must be at least 1 pigeonhole that has more than one pigeon in it.
- If k+1 or more objects are placed into k drawers, then there is at least one drawer containing two or more of the objects.

Pigeonhole Principle Examples

- In a group of 367 people, there must be two people with the same birthday.
  - There are 366 possible birthdays.

- In a group of 27 English words, at least two words must start with the same letter.
  - There are 26 letters.

Generalized Pigeonhole Principle

If N pigeons fly into k pigeonholes, then there is at least one pigeonhole containing $\lceil N/k \rceil$ pigeons.

Why is this true?

Solution

Let P denote the maximum number of pigeons in a pigeonhole.

The average number of pigeons in a pigeonhole is

$$N/k.$$

P must be at least as great as the average, i.e.,

$$P \geq N/k.$$

But P is an integer, therefore,

$$P \geq \lceil N/k \rceil.$$

Joke only programmers will get

Why did the functions stop calling each other?

Ans: Because they had too many arguments.

Relations

Textbook Reading:

Chapter 3, pp. 157-174.

Binary Relations

- Binary relations represent relationships between the elements of two sets.
- A **binary relation** R on set A and B is defined by: $R \subseteq A \times B$
- If $(a,b) \in R$, we write:

  aRb (a is related to b by R)

- If $(a,b) \notin R$, we write:

  $a \not{R} b$ (a is not related to b by R)

Binary Relations

- A binary relation is represented by a *set* of *ordered pairs.*
- If A = {a, b} and B = {1, 2, 3}, then a relation R_1 from A to B might be, for example, $R_1 = \{(a,2), (a,3), (b,2)\}$.
- The first element in each ordered pair comes from set A, and the second element in each ordered pair comes from set B.

Example

$A = \{0,1,2\}$

$B = \{a,b\}$

$A \times B = \{(0,a), (0,b), (1,a), (1,b), (2,a), (2,b)\}$

Then $R = \{(0,a), (0,b), (1,a), (2,b)\}$ is a *relation* from A to B.

- ✓Can we write $0Ra$? *yes*
- ✓Can we write $2Rb$? *yes*
- ✓Can we write $1Rb$? *no*

Example

- A binary relation may be represented graphically or as a table:

$$\begin{tikzcd}
0 \arrow[r] \arrow[dr] & a \\
1 \arrow[ur] & b \\
2 \arrow[ur] &
\end{tikzcd}$$

R	a	b
0	X	X
1	X	
2		X

We can see that *0Ra* but $1 \not R b$.

Functions as Binary Relations

- A function is a binary relation that has the restriction that each element of *A* can be related to exactly one element of *B*.

Relation

$$\begin{tikzcd}1 \arrow[r] \arrow[dr] & a \\ & b\end{tikzcd}$$

Function

$$\begin{tikzcd}1 \arrow[r] & a \\ & b\end{tikzcd}$$

Inverse Binary Relation

The inverse binary relation of R denote R^{-1} is

$$\{(x,y):(y,x)\in R\}$$

PSN. Find the inverse relation of

{(1,1), (1,2), (1,3), (1,4), (2,2), (2,3), (2,4), (3,3), (3,4), (4,4)}

Number of Binary Relations

How many binary relations are there for an *m* element set *A* and *n* element set *B*?

A binary relation set *A* and *B* is a subset of $A \times B$. Therefore the set of all relations is $P(A \times B)$, the power set of $A \times B$.

$$|P(A \times B)| = 2^{|A \times B|}$$

Now

$$|A \times B| = |A| \times |B| = mn$$

Therefore,

$$|P(A \times B)| = 2^{|A \times B|} = 2^{mn}.$$

Binary Relations on a Set

- Relations can also be from a set to itself.
- A relation on the set A is a relation from set A to set A, i.e., $R \subseteq A \times A$
- Let $A = \{1, 2, 3, 4\}$
- Which ordered pairs are in the relation $R = \{(a,b) \mid a \text{ divides } b\}$?
- $R = \{(1,1), (1,2), (1,3), (1,4), (2,2), (2,4), (3,3), (4,4)\}$

Binary Relations on a Set, cont'd

- Which of these relations (on the set of integers) contain each of the pairs (1,1), (1,2), (2,1), (1,-1), and (2,2)?

$$R_1 = \{(a,b) \mid a \leq b\}$$
$$R_2 = \{(a,b) \mid a > b\}$$
$$R_3 = \{(a,b) \mid a = b, a = -b\}$$
$$R_4 = \{(a,b) \mid a = b\}$$
$$R_5 = \{(a,b) \mid a = b + 1\}$$
$$R_6 = \{(a,b) \mid a + b \leq 3\}$$

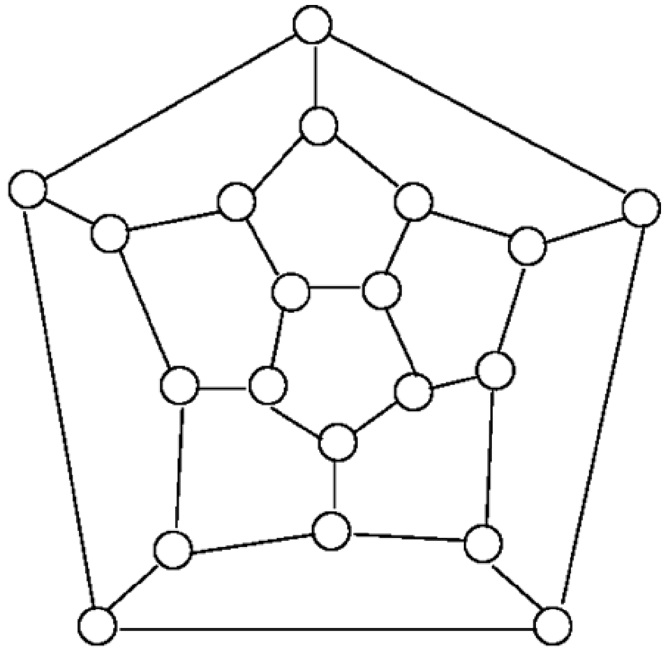
Binary Relations on a Set, cont'd

$$
\begin{aligned}
R_1 &= \{(a,b) \mid a \le b\} \\
R_2 &= \{(a,b) \mid a > b\} \\
R_3 &= \{(a,b) \mid a = b, a = -b\} \\
R_4 &= \{(a,b) \mid a = b\} \\
R_5 &= \{(a,b) \mid a = b + 1\} \\
R_6 &= \{(a,b) \mid a + b \le 3\}
\end{aligned}
$$

- The pair (1,1) is in R_1, R_3, R_4 and R_6
- The pair (1,2) is in R_1 and R_6
- The pair (2,1) is in R_2 , R_5 and R_6
- The pair (1,-1) is in R_2 , R_3 and R_6
- The pair (2,2) is in R_1 , R_3 and R_4

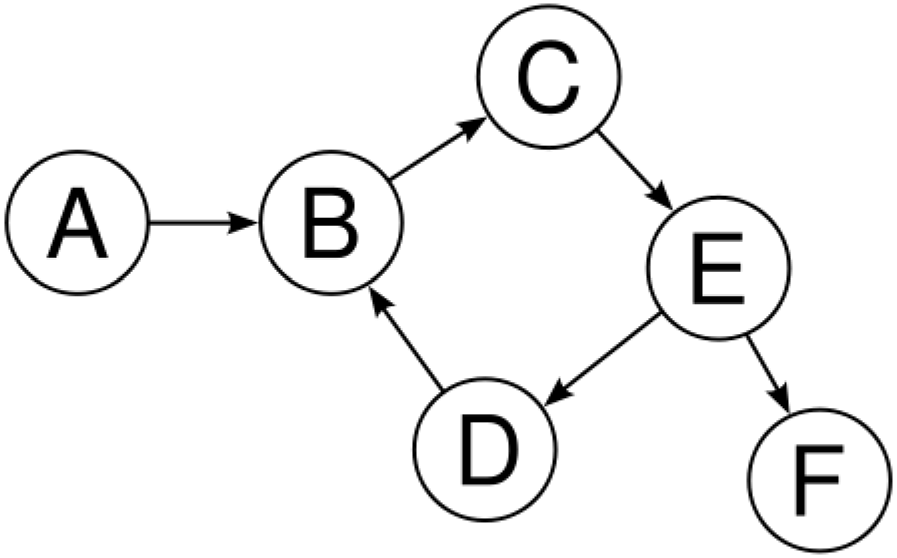
Graphs

The edge set of a graph determines a symmetric binary relation on the set of vertices called an adjacency relation.



Relation on set *V* corresponds to the edge set of a digraph with vertex set *V*

Relation {(A,B),(B,C),(C,D),(E,D),(E,F)} on set {A,B,C,D,E,F}



Number of Relations on a Set

How many relations are there on a set with n elements?

A relation on a set A is a subset of $A \times A$. Therefore the set of all relations is $P(A \times A)$, the power set of $A \times A$.

$$|P(A \times A)| = 2^{|A \times A|}$$

Now

$$|A \times A| = |A| \times |A| = n \times n = n^2$$

Therefore,

$$|P(A \times A)| = 2^{|A \times A|} = 2^{n^2}.$$

Example

- How many relations are there on set $S = \{a, b, c\}$?
- There are 3 elements in set S, so $S \times S$ has $3^2 = 9$ elements.
- Therefore, there are $2^9 = 512$ different relations on the set $S = \{a, b, c\}$.

Reflexive

- Let R be a relation on set A.
- R is **reflexive** if:

  $(a, a) \in R$ for every element $a \in A$.

Reflexive cont'd

- Which of these is reflexive?

R_1 = {(1,1), (1,2), (2,1), (2,2), (3,4), (4,1), (4,4)}
R_2 = {(1,1), (1,2), (2,1)}
R_3 = {**(1,1)**, (1,2), (1,4), (2,1), **(2,2)**, **(3,3)**, (4,1), **(4,4)**}
R_4 = {(2,1), (3,1), (3,2), (4,1), (4,2), (4,3)}
R_5 = {**(1,1)**, (1,2), (1,3), (1,4), **(2,2)**, (2,3), (2,4), **(3,3)**, (3,4), **(4,4)**}
R_6 = {(3,4)}

- The relations R_3 and R_5 are reflexive because they contain **all** pairs of the form (a,a); the other don't [they are all missing (3,3)].

Symmetric

- Let R be a relation on set A.
- R is **symmetric** if:

  $(b, a) \in R$ whenever $(a, b) \in R$,

  where $a, b \in A$.

A relation is symmetric iff “a is related to b” implies that “b is related to a”.

Symmetric cont'd

- Which of these is symmetric?

$R_1 = \{(1,1), (1,2), (2,1), (2,2), (3,4), (4,1), (4,4)\}$
$R_2 = \{(1,1), (\mathbf{1,2}), (\mathbf{2,1})\}$
$R_3 = \{(1,1), (\mathbf{1,2}), (\mathbf{1,4}), (\mathbf{2,1}), (2,2), (3,3), (\mathbf{4,1}), (4,4)\}$
$R_4 = \{(2,1), (3,1), (3,2), (4,1), (4,2), (4,3)\}$
$R_5 = \{(1,1), (1,2), (1,3), (1,4), (2,2), (2,3), (2,4), (3,3), (3,4), (4,4)\}$
$R_6 = \{(3,4)\}$

- The relations R_2 and R_3 are symmetric because in each case (b,a) belongs to the relation whenever (a,b) does.
- The other relations aren't symmetric.

Antisymmetric

- Let R be a relation on set A.
- R is **antisymmetric** if whenever $(a, b) \in R$ and $(b, a) \in R$, then $a = b$, where $a, b \in A$.
- A relation is antisymmetric iff there are no pairs of distinct elements with a related to b and b related to a. That is, the only way to have a related to b and b related to a is for a and b to be the same element.
- Symmetric and antisymmetric are NOT exactly opposites.

Asymmetric vs. Antisymmetric

- Let R be a relation on set A.
- R is **asymmetric** if $(a, b) \in R$ implies $(b,a) \notin R$
- The relation < on the set of real numbers is asymmetric.
- R is **antisymmetric** if $(a, b) \in R$ and $(b,a) \in R$ implies $a = b$.
- The relation ≤ on the set of real numbers is antisymmetric.

Example

- Which of these is antisymmetric?

$R_1 = \{(1,1), (1,2), (2,1), (2,2), (3,4), (4,1), (4,4)\}$
$R_2 = \{(1,1), (1,2), (2,1)\}$
$R_3 = \{(1,1), (1,2), (1,4), (2,1), (2,2), (3,3), (4,1), (4,4)\}$
$R_4 = \{(2,1), (3,1), (3,2), (4,1), (4,2), (4,3)\}$
$R_5 = \{(1,1), (1,2), (1,3), (1,4), (2,2), (2,3), (2,4), (3,3), (3,4), (4,4)\}$
$R_6 = \{(3,4)\}$

- The relations R_4, R_5 and R_6 are antisymmetric because there is no pair of elements a and b with $a \neq b$ such that both (a,b) and (b,a) belong to the relation.

- The other relations aren't antisymmetric.

Transitivity

Let R be a relation on set A.

R is **transitive** if whenever $(a ,b) \in R$ and $(b, c) \in R$, then $(a, c) \in R$, where $a, b, c \in A$.

Example

- Which of these is transitive?

$R_1 = \{(1,1), (1,2), (2,1), (2,2), (3,4), (4,1), (4,4)\}$
$R_2 = \{(1,1), (1,2), (2,1)\}$
$R_3 = \{(1,1), (1,2), (1,4), (2,1), (2,2), (3,3), (4,1), (4,4)\}$
$R_4 = \{ (2,1), (3,1), (3,2), (4,1), (4,2) , (4,3)\}$
$R_5 = \{(1,1), (1,2), (1,3), (1,4), (2,2), (2,3), (2,4), (3,3), (3,4), (4,4)\}$
$R_6 = \{(3,4)\}$

- The relations R_4, R_5 and R_6 are transitive because if (a,b) and (b,c) belong to the relation, then (a,c) does also.
- The other relations aren't transitive.

Combining Relations

Relations from A to B are subsets of $A \times B$.

For example, if $A = \{1, 2\}$ and $B = \{a, b\}$, then $A \times B = \{(1, a), (1, b), (2, a), (2, b)\}$

Two relations from A to B can be combined in any way that two sets can be combined. Specifically, we can find the *union*, *intersection*, *exclusive-or*, and *difference* of the two relations.

Combining Relations cont'd

Let A = {1, 2, 3} and B= {1, 2, 3, 4}, and suppose we have the relations:

$R_1 = \{(1,1), (2,2), (3,3)\}$

$R_2 = \{(1,1), (1,2), (1,3), (1,4)\}$.

Then we can find the union, intersection, and difference of the relations:

$R_1 \cup R_2 = \{(1,1), (1,2), (1,3), (1,4), (2,2), (3,3)\}$

$R_1 \cap R_2 = \{(1,1)\}$

$R_1 - R_2 = \{(2,2), (3,3)\}$

$R_2 - R_1 = \{(1,2), (1,3), (1,4)\}$

Composition of Relations

- Composition of relations generalizes composition of functions.
- If R_1 is a relation from A to B and R_2 is a relation from B to C, then the composition of R_1 with R_2 (denoted $R_2 \circ R_1$) is the relation from A to C
- It is defined by: (a, b) is a member of R_1 and (b, c) is a member of R_2, then (a, c) is a member of $R_2 \circ R_1$, where $a \in A$, $b \in B$, $c \in C$.

Example

- Let $A=\{1,2,3\}$, $B=\{w,x,y,z\}$, $C=\{A,B,C,D\}$

  $R_1=\{(1,z),(2,w)\}$, $R_2=\{(w,B),(w,D),(x,A)\}$
- Find $R_2 \circ R_1$
- **Match $(a,b) \in R_1$ with $(b,c) \in R_2$ to get $(a,c) \in R_2 \circ R_1$**
- R_2's b's are w and x; R_1's b's are z and w
- Only the w's match; R_1 has only 1 w pair, $(2,w)$
- So the (a, c) pairs will include 2 from R_1 and B and D from R_2: $(2, B)$, $(2, D)$

PSN

Given the following relations, find $R \circ S$:

$R = \{(1,0),(2,0), (3,1), (3,2), (4,1)\}$

$S = \{(1,1), (1,4), (2,3), (3,1), (3,4)\}$

Construct the ordered pairs in $R \circ S$ as follows:

```
for each ordered pair $(s_1,s_2)$ in $S$
    for each ordered pair $(r_1,r_2)$ in $R$
        if $s_2 = r_1$ then
            $(s_1,r_2)$ belongs to $R \circ S$
```

Inverse of Composition of Relations

$$(R \circ S)^{-1} = S^{-1} \circ R^{-1}$$

Example

R = {(1,0),(2,0), (3,1), (3,2), (4,1)}

S = {(1,1), (1,4), (2,3), (3,1), (3,4)}

$R \circ S$ = {(1,0), (1,1), (2,1), (2,2), (3,0), (3,1)}

R^{-1} = {(0,1),(0,2), (1,3), (2,3), (1,4)}

S^{-1} = {(1,1), (4,1), (3,2), (1,3), (4,3)}

$(R \circ S)^{-1}$ = {(0,1), (1,1), (1,2), (2,2), (0,3), (1,3)} $= S^{-1} \circ R^{-1}$

The Powers of a Relation

- The powers of a relation R are recursively defined from the definition of a composite of two relations.

- Let R be a relation on the set A. The powers R^n, for $n = 1, 2, 3, \ldots$ are defined recursively by:

  $R^1 = R$

  $R^{n+1} = R^n \circ R$:

So:

$R^2 = R \circ R$

$R^3 = R^2 \circ R = (R \circ R) \circ R$

etc.

PSN. Let $R = \{(1,1), (2,1), (3,2), (4,3)\}$

Find the powers R^n , where $n = 1, 2, 3, 4, 5$

Transitivity

It follows from the definition of transitivity that

A relation *R* on a set *A* is transitive iff

$R^n \subseteq R$ for n = 1, 2, 3, 4, …

For example, we showed the following relationship was transitive

R = {(1,1), (1,2), (1,3), (1,4), (2,2), (2,3),
(2,4), (3,3), (3,4), (4,4)}

Check that $R^2 = R$.

Transitive Closure

- Let *R* be a relation on a set *A* of size *n.*
- The **transitive closure** of *R* is obtained by repeatedly adding the pair (x,z) whenever there is a pair (x,y) and (y,z) until for every pair (x,y) and (y,z) the pair (x,z) is in the relation.
- For example, if *A* = {a,b,c} and *R* = {(a,b),(b,c),(c,d)}, then the transitive closure is {(a,b),(b,c),(c,d),(a,c),(b,d),(a,d)}.
- The transitive closure equals $R^1 \cup R^2 \cup \cdots \cup R^{n-1}$.
- If *R* is reflexive then $R^{i-1} \subseteq R^i, i = 2, 3, \dots, n - 1$. It follows that R^{n-1} is the transitive closure.

Why was the cell phone wearing glasses?

Answer. It lost its contacts.

Equivalence Relations and Partial Orders

Textbook reading:

Chapter 3, Section 3.6, pp. 181-187

Section 3.8, pp. 191-195

Equivalence Relations

A relation on set A is called an *equivalence relation* if it is:

- reflexive
- symmetric, and
- transitive

Equivalence Relations

Two elements a and b that are related by an equivalence relation are said to be *equivalent*.

Example – Equivalence Relation

Let R be a relation on set A, where $A = \{1, 2, 3, 4, 5\}$ and $R = \{(1,1), (2,2), (3,3), (4,4), (5,5), (1,3), (3,1)\}$

Is R an equivalence relation?

- *Reflexive* – it contains$\{(1,1), (2,2), (3,3), (4,4), (5,5)$
- *Symmetric* – it contains both (1,3) and (3,1)
- *Transitive* – for each pair of pairs (x,y) and (y, z) in R, the pair (x,z) is also in R.

Yes, it is an equivalence relation.

Example – Congruence modulo m

Let $R = \{(a, b) \mid a \equiv b \pmod{m}\}$ be a relation on the set of integers and m be a positive integer > 1.

Is R an equivalence relation?

- *Reflexive* – is it true that $a \equiv a \pmod{m}\}$? *yes*
- *Symmetric* – is it true that if $a \equiv b \pmod{m}$ then $b \equiv a \pmod{m}$? *yes*
- *Transitive* - is it true that if $a \equiv b \pmod{m}$ and $b \equiv c \pmod{m}$ then $a \equiv c \pmod{m}$? *yes*

Example – Strings

R is the relation on the set of strings of English letters such that *aRb* iff $l(a) = l(b)$, where $l(x)$ is the length of the string *x*.

Is *R* an equivalence relation?

Example – Strings cont'd

Since $l(a) = l(a)$, then aRa for any string a. So R is reflexive.

Suppose aRb, so that $l(a) = l(b)$. Then it is also true that $l(b) = l(a)$, which means that bRa. Consequently, R is symmetric.

Suppose aRb and bRc. Then $l(a) = l(b)$ and $l(b) = l(c)$. Therefore, $l(a) = l(c)$ and so aRc. Hence, R is transitive.

Therefore, R is an equivalence relation.

Equivalence Classes

Let R be a equivalence relation on set A.

The set of all elements that are related to an element a of A is called the *equivalence class* of a.

The equivalence class of a with respect to R is:

$$[a]_R = \{s \mid (s, a) \in R\}$$

- When only one relation is under consideration, we will just write $[a]$.

Equivalence Classes cont'd

If R is a equivalence relation on a set A, the *equivalence class* of the element a is:

$$[a]_R = \{s \mid (s, a) \in R\}$$

If $b \in [a]_R$, then b is called a *representative* of this equivalence class.

Equivalence Classes – Example 1

Let R be the relation on the set of integers such that aRb iff $a = b$ or $a = -b$. We can show that this is an equivalence relation.

The equivalence class of element a is

$$[a] = \{a, -a\}$$

Examples:

$$[7] = \{7, -7\}$$
$$[-5] = \{5, -5\}$$
$$[0] = \{0\}$$

Equivalence Classes – Example 2

Consider the equivalence relation R on set A. What are the equivalence classes?

$$A = \{1, 2, 3, 4, 5\}$$

$$R = \{(1,1), (2,2), (3,3), (4,4), (5,5), (1,3), (3,1)\}$$

Just look at the aRb relationships. Which elements are related to which?

$[1] = \{1, 3\}$ $[2] = \{2\}$

$[3] = \{3, 1\}$ $[4] = \{4\}$

$[5] = \{5\}$

[1] and [3] are the same equivalence classes.

Equivalence Classes – Example 3

Consider set

A = {“hello”, “world”, “CS”, “discrete”, “Hi”, “joe”, “text”, “math”, “sting”, “purple”, “doe”, “bye”}.

Equivalence class on A for relation *aRb* whenever *a* and *b* have the same size:

[“hello”] = {“hello”, “world”, “sting”}

[“math”] = {“text”, “math”}

[“CS”] = {“CS”, “Hi”}

[“discrete”] = {“discrete”}

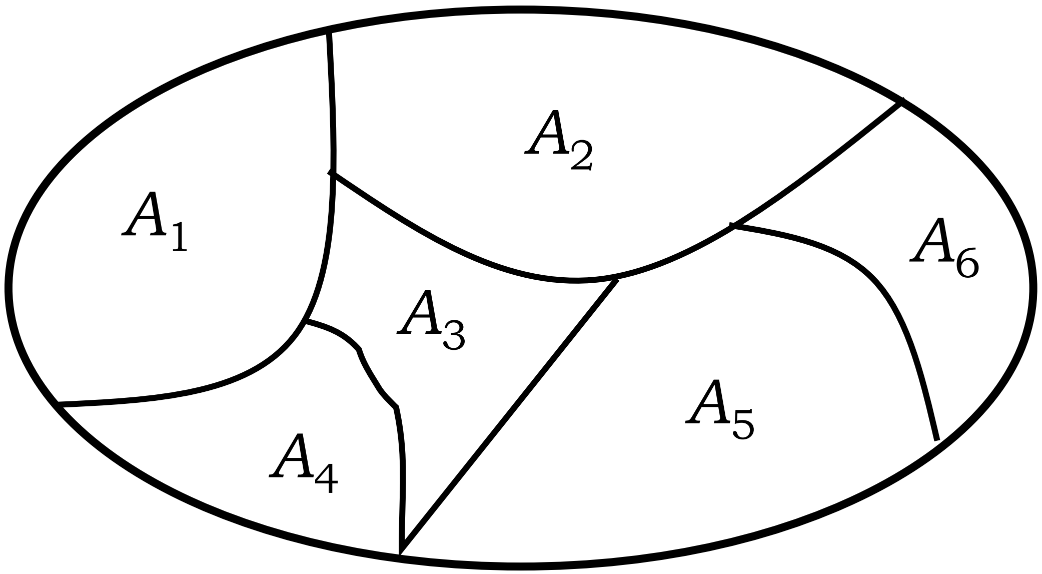
[“joe”] = {“joe”, “doe”, “bye”}

[“purple”] = {“purple”}

Partitions

A **partition** of a set *A* divides *A* into non-overlapping subsets:

- A partition of a set A is a collection of disjoint nonempty subsets of A that have A as their union.



Set *A*

Partitions – Example 1

$S = \{a, b, c, d, e, f\}$

$S_1 = \{a, d, e\}$

$S_2 = \{b\}$

$S_3 = \{c, f\}$

$P = \{S_1, S_2, S_3\}$

P is a partition of set S

Partitions – Example 2

If $S = \{1, 2, 3, 4, 5, 6\}$, then

$A_1 = \{1, 3, 4\}$

$A_2 = \{2, 5\}$

$A_3 = \{6\}$

form a partition of S, because:

- these sets are disjoint
- the union of these sets is S.

Violating Partition Property – Not Disjoint

$S = \{1, 2, 3, 4, 5, 6\}$

$A_1 = \{1, 3, 4, 5\}$

$A_2 = \{2, 5\}$

$A_3 = \{6\}$

Does not form a partition of *S*, because these sets are not disjoint (5 occurs in two different sets)

Violating Partition Property – Union is not *S*

$S = \{1, 2, 3, 4, 5, 6\}$

$A_1 = \{1, 3\}$

$A_2 = \{2, 5\}$

$A_3 = \{6\}$

Do **not** form a partition of *S*, because the union of these sets is not *S* (since 4 is not a member of any of the subsets, but is a member of *S*).

Violating Partition Property – element not in S

If $S = \{1, 2, 3, 4, 5, 6\}$, then

$A_1 = \{1, 3, 4\}$

$A_2 = \{2, 5\}$

$A_3 = \{6, 7\}$

Do **not** form a partition of S, because 7 is a member of set A_3 but is not a member of S.

Partitions and Equivalence Relations

- Let R be an equivalence relation on set S
- Then the equivalence classes of R form a partition of S.
- Conversely, let $P = \{A_i \mid i \in I\}$ be a partition of set S.
- Then there is an equivalence relation R that has the sets A_i $(i \in I)$ as its equivalence classes.

PSN. Define the equivalence relation R corresponding to P and prove its an equivalence relation.

Constructing an Equivalence Relation from a Partition

Given set $S = \{1, 2, 3, 4, 5, 6\}$ and a partition of S

$A_1 = \{1, 2, 3\}$

$A_2 = \{4, 5\}$

$A_3 = \{6\}$

find the ordered pairs that make up the equivalence relation R produced by that partition.

Constructing an Equivalence Relation from a Partition

Let's find the ordered pairs that are in R:

$A_1 = \{1, 2, 3\} \rightarrow (1,1), (1,2), (1,3), (2,1), (2,2),$
$(2,3), (3,1), (3,2), (3,3)$

$A_2 = \{4, 5\} \rightarrow (4,4), (4,5), (5,4), (5,5)$

$A_3 = \{6\} \rightarrow (6,6)$

So R is just the set consisting of all these ordered pairs:

$R = \{(1,1), (1,2), (1,3), (2,1), (2,2), (2,3), (3,1), (3,2),$
$(3,3), (4,4), (4,5), (5,4), (5,5), (6,6)\}$

Partial Order

A relation R on a set S is called a partial ordering or *partial order* if it is:

- reflexive
- antisymmetric
- transitive

Partially Ordered Set or Poset

A set S together with a partial ordering R is called a **partially ordered set**, or **poset**, and is denoted by (S, R).

Example – Poset

Let R be a relation on set A. Is R a partial order?

$$A = \{1, 2, 3, 4\}$$

$$R = \{(1,1), (1,2), (1,3), (1,4), (2,2),$$
$$(2,3), (2,4), (3,3), (3,4), (4,4)\}$$

Example – Poset cont'd

$R = \{(1,1), (1,2), (1,3), (1,4), (2,2), (2,3), (2,4),$

$(3,3), (3,4), (4,4)\}$

To be a partial order, R must be reflexive, antisymmetric, and transitive.

R is reflexive because R includes (1,1), (2,2), (3,3), (4,4).

R is antisymmetric because for every pair (a,b) in R, (b,a) is not in R (unless $a = b$).

R is transitive because for every pair (a,b) in R, if (b,c) is in R then (a,c) is also in R.

Example – Poset cont'd

So, given

$A = \{1, 2, 3, 4\}$

$R = \{(1,1), (1,2), (1,3), (1,4), (2,2),$
$(2,3), (2,4), (3,3), (3,4), (4,4)\}$

R **is** a partial order, and (*A*, *R*) **is** a poset.

Second Example Poset

Is the "≥" relation a partial ordering on the set of integers?

- Since $a \geq a$ for every integer a, ≥ is reflexive
- If $a \geq b$ and $b \geq a$, then $a = b$. Hence ≥ is anti-symmetric.
- Since $a \geq b$ and $b \geq c$ implies $a \geq c$, is transitive.
- Therefore "≥" is a partial ordering on the set of integers and (Z, ≥) is a poset.

$\subseteq$ determine a partial order on sets

Consider the power $P(A)$ of A, i.e., $P(A)$ is the collection of all subsets of A. Then, the $\subseteq$ relation determines a partial ordering on $P(A)$ and $(P(A), \subseteq)$ is a poset.

It is an easy exercise to verify that $\subseteq$ is reflexive, antisymmetric and transitive.

Comparable / Incomparable

In a poset the notation $a \preccurlyeq b$ denotes $(a, b) \in R$

The "less than or equal to" ($\leq$) is an example of partial ordering

The elements a and b of a poset $(S, \preccurlyeq)$ are called *comparable* if either $a \preccurlyeq b$ or $b \preccurlyeq a$.

The elements a and b of a poset $(S, \preccurlyeq)$ are called *incomparable* if neither $a \preccurlyeq b$ nor $b \preccurlyeq a$.

In the poset $(Z^+, |)$ where $|$ means divides:

- Are 3 and 9 comparable? *Yes; 3 divides 9*
- Are 5 and 7 comparable? *No; neither divides the other*

In the poset $(P(\{1,2,3,4,5\}), \subseteq)$

- Are $\{2,5\}$ and $\{1,2,4,5\}$ comparable? *Yes;* $\{2,5\} \subseteq \{1,2,4,5\}$
- Are $\{2,3,5\}$ and $\{1,2,4,5\}$ comparable? *No; neither is a subset of the other*

Linear Order or Total Order

We said: “Partial ordering” because pairs of elements may be incomparable.

If every two elements of a poset $(S, \preccurlyeq)$ are comparable, then S is called a *totally ordered* or *linearly ordered* set and $\preccurlyeq$ is called a *total order* or *linear order*.

A totally ordered set is also called a *chain*.

Total Order

The poset $(Z, \leq)$ is totally ordered. Why?

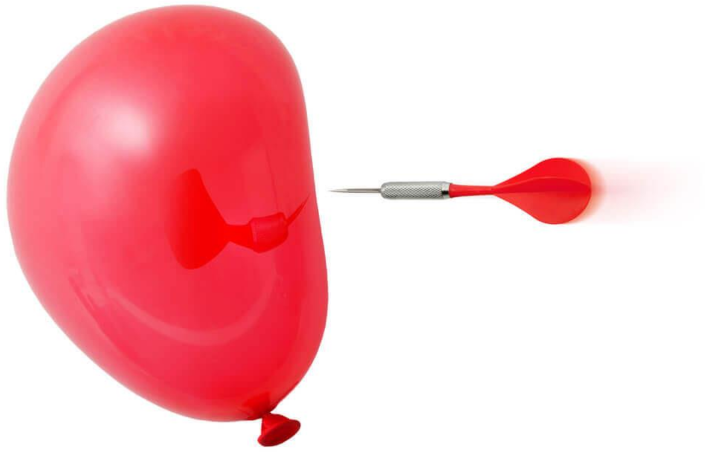
Every two elements of **Z** are comparable; that is, $a \leq b$ or $b \leq a$ for all integers.

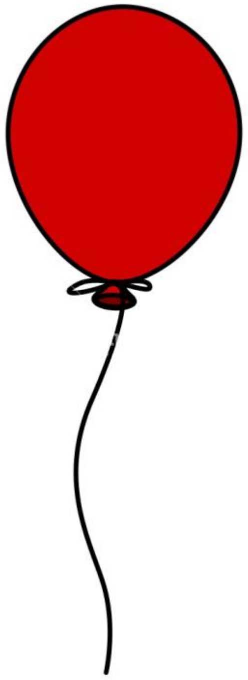
The poset $(Z^+, |)$ is not totally ordered where $|$ means divides. Why?

It contains elements that are incomparable; for example $5 \nmid 7$.

What's a balloon's least favorite type of music?

Pop





Modular Arithmetic

Given integers *n* and *k*, upon dividing *n* by *k*, we obtain a quotient *q* and remainder *r* given by

$$n = kq + r$$

We define *n* modulo *k* or simply ***n* mod *k*** to be the **remainder** *r.*

For example,

208 mod 10 = 8 since 208 = 20×10 + 8
45 mod 6 = 3 since 45 = 6×7 + 3
108 mod 13 = 4 since 108 = 13×8 + 4

Modular Arithmetic

If both *x* and *y* have the same remainder upon dividing by *n*, we write

$$x \equiv y \pmod{n}$$

Proposition. $x \equiv y \pmod{n}$ iff $x - y$ is divisible by *n*.

For example,

$208 \equiv 188 \pmod{10}$. $208 - 108 = 10\times10$ is divisible by 10

$40 \equiv 14 \pmod{13}$. $40 - 14 = 26 = 2\times13$ is divisible by 13

$206 \equiv 342 \pmod{17}$. $206 - 342 = -136 = -8\times17$ is divisible by 17.

Two important properties

$(x + y)$ **mod** $k = ((x$ **mod** $k) + (y$ **mod** $k))$ **mod** k

$(x \times y)$ **mod** $k = ((x$ **mod** $k) \times (y$ **mod** $k))$ **mod** k

It follows that we can compute an expression mod *k* where the expression is obtained by performing a sequence of additions and multiplications by reducing the result mod *k* after each operation is performed. Thus, an expression involving a large integer mod *k* can be computed by reducing the result of each computation mod *k*. This is important in cryptographic applications, which often involve integers having hundreds, even thousands, of digits.

Equivalence Relation

PSN. Show that the relation R given by xRy whenever $x \equiv y \pmod{n}$, i.e., whenever n divides $x - y$, is an equivalence relation on the set of Z of integers.

Equivalence Classes

with equivalence classes:

$$[x] = \{\ldots, x - 2n, x - n, x, x + n, x + 2n, \ldots\}$$

of all integers y such that $x \equiv y \pmod{n}$.

Residues mod *n*

$Z_n = \{0, \ldots, n-1\}$ of integers (or residues) mod *n* are defined the same as over the integers, but the result *x* of each operation is reduced by replacing *x* with the remainder *r* when *x* is divided by *n.*

Element $x \in \{0,1,2,\ldots,n-1\}$ corresponds to the equivalence class

$$[x] = \{\ldots, x-2n, x-n, x, x+n, x+2n, \ldots\}$$

That is, we identify the integers 0, 1, 2, …, $n-1$ with their equivalence classes [0], [1], [2], …, $[n-1]$.

Example: integers mod 4

$$Z_4 = \{0, 1, 2, 3\}$$

Elements correspond to classes:

$[0] = \{\ldots, -8, -4, 0, 4, 8, \ldots\}$

$[1] = \{\ldots, -7, -3, 1, 5, 9, \ldots\}$

$[2] = \{\ldots, -6, -2, 2, 6, 10, \ldots\}$

$[3] = \{\ldots, -5, -1, 3, 7, 11, \ldots\}$

Addition and Multiplication Tables:

+	**0**	**1**	**2**	**3**
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

×	**0**	**1**	**2**	**3**
0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1

PSN. Obtain addition and multiplication table for $n = 7$.

Z_n forms a commutative ring

It is easily verified that Z_n satisfies the following commutative ring properties:

Addition is commutative, associative and every element has an inverse, so that Z_n is a commutative (Abelian) group under addition:

$$x + y \equiv y + x \pmod{n}$$

$$(x + y) + z \equiv x + (y + z) \pmod{n}$$

$$x + (-x) \equiv 0 \pmod{n}$$

Multiplication is commutative and associative:

$$x * y \equiv y * x \pmod{n}$$

$$(x * y) * z \equiv x * (y * z) \pmod{n}$$

Multiplication distributes over addition:

$$x*(y + z) \equiv x*y + x*z \pmod{n}$$

Z_n forms a field for n prime

In the case when n is prime, Z_n is also a commutative group under multiplication, so that it determines a field known as the **Galois field of integers modulo *n***, denoted by GF(n).

Évariste Galois

Fermat's Little Theorem

Theorem (Fermat). Let *b* and *n* be positive integers, where *n* is prime and *b* is not divisible by *n*. Then,

$$b^{n-1} \equiv 1 \pmod{n}.$$

Pierre de Fermat
(1601-1665)

Proof of Fermat's Little Theorem

$$Z_n - \{0\} = \{1, 2, .., n - 1\}$$

Let f: $Z_n - \{0\} \to Z_n - \{0\}$ be defined by

$$f(i) = b \times i \pmod{n}, i \in Z_n - \{0\}$$

Proposition. f is bijective.

Corollary. $\{f(1), f(2), .., f(n - 1)\}$ = $\{1, 2, .., n - 1\}$,

where $f(i)$ is reduce to a residue mod *n*.

Illustration of Corollary

$\underline{n = 7, b = 2}$

$$\{2 \times 1, 2 \times 2, 2 \times 3, 2 \times 4, 2 \times 5, 2 \times 6\} = \{2, 4, 6, 1, 3, 5\}$$

$\underline{n = 7, b = 5}$

$$\{5 \times 1, 5 \times 2, 5 \times 3, 5 \times 4, 5 \times 5, 5 \times 6\} = \{5, 3, 1, 6, 4, 2\}$$

$\underline{n = 7, b = 6}$

$$\{6 \times 1, 6 \times 2, 6 \times 3, 6 \times 4, 6 \times 5, 6 \times 6\} = \{6, 5, 4, 3, 2, 1\}$$

Proof of Fermat's Little Theorem, cont'd

We first show that f is injective (1-1).

Suppose $f(i) = f(j)$

Then,

$b \times i \equiv b \times j \pmod{n}$

$\Rightarrow b^{-1}b \times i \equiv b^{-1}b \times j \pmod{n}$

$\Rightarrow i = j$

This proves f is injective. A mapping from a finite set to itself, which is injective, must necessarily be surjective, i.e., onto, and therefore, bijective. This can be proved by contradiction as follows.

Assume f is an injective (1-1) mapping from Z_n to Z_n, but not surjective (onto). Then, the range R is not equal to the whole set Z_n, i.e., $R \subset Z_n$. But since f is 1-1, $Z_n = R$. This, implies that $Z_n \subset Z_n$, a contradiction.

Proof of Fermat’s Little Theorem cont’d

It follows from the Corollary that

$$(b \times 1) \times (b \times 2) \times \cdots \times (b \times n - 1) \equiv 1 \times 2 \times \cdots \times n - 1 \pmod{n}$$

Therefore we have

$$\Rightarrow b^{n-1} \times (1 \times 2 \times \cdots \times n - 1) \equiv 1 \times 2 \times \cdots \times n - 1 \quad \pmod{n}$$

$$\Rightarrow b^{n-1} \equiv 1 \pmod{n}$$

Q.E.D. ("quod erat demonstrandum", Latin for "that which was to be demonstrated“)

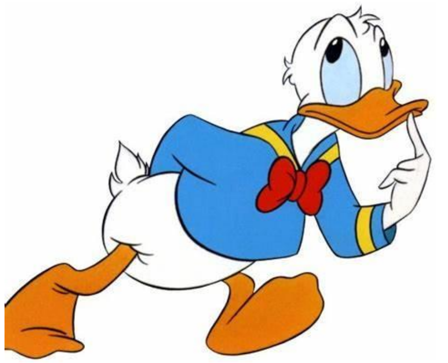
Example $n = 7$, $b = 2$

$$(2\times1) \times (2\times2) \times (2\times3) \times (2\times4) \times (2\times5) \times (2\times6) \equiv 2 \times 4 \times 6 \times 1 \times 3 \times 5 \pmod 7$$

$$\Rightarrow 2^6 \times (1 \times 2 \times 3 \times 4 \times 5 \times 6) \equiv 1 \times 2 \times 3 \times 4 \times 5 \times 6 \pmod 7$$

$$\Rightarrow 2^6 \equiv 1 \pmod 7$$

You know what's odd?



Any integer not wholly divisible by 2.

Integers – Bases

We discuss

- Expressing a number in binary and more generally in base *b.*
- Designing a recursive algorithm to convert a decimal number to binary and more generally to base *b.*
- Relationship between a number and the number of its digits.

Binary Representation

The binary representation is a number *n* is

$$\boldsymbol{d_{k-1} d_{k-2} \cdots d_1 d_0}$$

where $d_i \in \{0,1\}, i = 0, \ldots, k$, such that

$$\boldsymbol{n = \ d_{k-1} \times 2^{k\ -1}\ d_{k-2} \times 2^{k\ -2} + \cdots d_{k-2} \times 2 + d_0}$$

For example, convert 114 to binary

114 = 64 + 32 + 16 + 2

Binary representation is

1110010

PSN. Convert 250 to binary.

Base *b*

Give a positive integers *b* and *n, n* is represented in base *b* as

$$d_{k-1}d_{k-2}\cdots d_1 d_0$$

where $0 \leq d_i \leq b-1, i = 0, \ldots, k, d_{k-1} \neq 0$, such that

$$n = \ d_{k-1} \times b^{k\,-1}\, d_{k-2} \times b^{k\,-2} + \cdots d_1 \times b + d_0$$

This representation is unique.

When *b* = 8 the representation is called octal and when *b* = 16 the representation is called hexadecimal.

Proof by contradiction that representation is unique

Suppose *n* could be represented in two ways, i.e.,

$$n = d_{k-1}d_{k-2}\cdots d_1 d_0 = e_{j-1}e_{j-2}\cdots e_1 e_0$$

Then,

$(d_{k-1}\times b^{k-1} + \cdots + d_1 \times b + d_0) - (d_{j-1}\times b^{j-1} + \cdots + d_1\times b + d_0)$ = 0

Suppose the highest power of *b* that doesn't get canceled out is *m*. Bring the term with b^m to one side (without loss of generality, we will assume the left-hand-side) and all other terms to the other side. Then

$$c \times b^m = c_0 + c_1 \times b + \cdots + c_{m-1} \times b^{m-1}$$

$c, c_0, \dots, c_{m-1}$, where $0 < c < b$ and $-b < c_i < b$ for $i = 0, 1, \dots, m-1$. It follows that

$$b^m \le (b-1) + (b-1)\times b + \cdots + (b-1)\times b^{m-1}$$

$$\le (b-1)(1 + b + \cdots + b^{m-1}) = \frac{(b-1)(b^m-1)}{b-1} = b^m - 1$$

We have $b^m \le b^m - 1$, a contradiction.

Converting to Binary Using Recursion

Give a recursive algorithm for changing from decimal to binary, i.e., base 2.

Solution

Use the fact that the least significant digit of *n* in its binary representation is

$$n \bmod 2$$

and the decimal number obtained by removing the least significant binary digit in its binary representation is

$$\left\lfloor \frac{n}{2} \right\rfloor$$

which is obtain by *n*/2 using integer division. For example, consider *n* = 225:

225 → 11100001

225 mod 2 = 1

225/2 = 112 → 1110000

Converting 114 to binary using recursion

$$114 \leftrightarrow \frac{114}{2}\ 114 \bmod 2 = 57\ 0$$

$$57\ 0 \leftrightarrow \left\lfloor\frac{57}{2}\right\rfloor\ 57 \bmod 2\quad 0 \leftrightarrow 28\ 1\ 0$$

$$28\ 1\ 0 \leftrightarrow \frac{28}{2}\ \ 28 \bmod 2\ \ 1\ 0 \leftrightarrow 14\ 0\ 1\ 0$$

$$14\ 0\ 1\ 0 \leftrightarrow \frac{14}{2}\ 14 \bmod 2\quad 0\ 1\ 0 \leftrightarrow 7\ 0\ 0\ 1\ 0$$

$$7\ 0\ 0\ 1\ 0 \leftrightarrow \left\lfloor\frac{7}{2}\right\rfloor\ \ 7 \bmod 2\ \ 0\ 0\ 1\ \ 0 \leftrightarrow 3\ 1\ 0\ 0\ 1\ 0$$

$$3\ 1\ 0\ 0\ 1\ 0 \leftrightarrow \left\lfloor\frac{3}{2}\right\rfloor\ \ 3 \bmod 2 \leftrightarrow 1\ 1\ 1\ 0\ 0\ 1\ 0$$

Binary representation is 1110010

PSN. Convert 250 to binary.

Recursive Function for Converting a Decimal number to Binary

Give pseudocode for a recursive function *BinRep*(*n*) for converting an number *n* to its binary (base 2) representation stored in a string. Assume + performs the operation of adding a digit, i.e.,

“10001101” + 0 → “100011010”

Convert to Binary

```
function BinRep(n) recursive
  Input:   n (a positive integer)
  Output: string for binary representation of n

    if $n == 0$ return (empty string)
    return( BinRep($n/2$) + $n$ mod 2 )

end BinRep
```

Note that *BinRep* works for *n* strictly positive. If *n* is zero, it returns the empty string. We need to code it this way, since otherwise a leading 0 will be added to all *n* greater than 0.

Changing to a General Base

Design a recursive algorithm for changing from decimal to any given base *b*.

Solution

Use the fact that the least significant digit of *n* in its binary representation is

$$n \bmod b$$

and the decimal number obtained by removing the least significant binary digit in its binary representation is

$$\left\lfloor \frac{n}{b} \right\rfloor$$

which is obtain by *n*/*b* using integer division.

Recursive Function for Converting a number of Binary

Give pseudocode for a recursive function *BinRep*(*n*) for converting an number *n* to its base *b* representation stored in a string. Assume + performs the operation of adding a digit. For example for *b* = 8

“20367471235” + 6 → “203677712356”

Convert to Base *b*

```
function BinRep(n,b) recursive
  Input:   n,b (positive integers)
  Output: string for representation of n in base b

    if n == 0 return (empty string)
    return( BinRep(n/b) + n mod b )

end BinRep
```

Note that *BinRep* works for *n* strictly positive. If *n* is zero, it returns the empty string. We need to code it this way, since otherwise a leading 0 will be added to all *n* greater than 0.

Relationship between a number and the number of its digits

In analyzing algorithms involving integers, the input size of an integer *n* is the number of digits of *n*.

A 100-digit number is enormous in the sense it is greater than the number of atoms in the known universe, which is estimated to be about 10^{83}, which as only 84 digits.

Yet a 100-digit number *n* has input size 100, which is relatively small and can be stored in an array of size 100.

Problem Solving Notebook

Show that the number *d* of decimal digits of *n* is approximately $\log_{10} n$.

What about number of binary digits? Octal digits? hexadecimal digits?

There are 10 kinds of people in the world.

Those who know binary and those who don't.

10 people

Greatest Common Divisor (gcd)

The greatest common divisor of two integers a and b, denoted by $\mathrm{gcd}(a, b)$ is the largest integer that divides both.

Computing gcd using prime factorization

The prime factorization of a number n is the unique product of prime powers that equals n.

For example,

$$3000 = 2^3 \times 3 \times 5^3$$

$$7700 = 2^2 \times 5^2 \times 7 \times 11$$

The gcd(a,b) is obtained the product of the of the smallest power of each prime from the prime factorizations of a and b, where the smallest power is 0 if the prime does not occur in the factorization.

$$\mathrm{gcd}(3000{,}7700) = \ 2^2 \times 5^2 = 100$$

- It turns out that prime factorization for large integers, i.e., hundreds of digits, is a “hard” problem and it is not known how to solve in real time.
- However, the greatest common divisor can be computed efficiently using an algorithm that dates all the way back to Euclid who lived c. 325 – c. 270 BC in Alexandria, Egypt.

Recurrence Relation for gcd

The key idea is to use the recursion relation

$\gcd(a, b) = \gcd(b, r)$, where $r = a \bmod b$.

The initial condition is $\gcd(a, 0) = a$.

The concept of 0 had not be invented in Euclid's time, so the initial condition he used was more cumbersome.

Example gcd(3000,7700)

$$3000 = 0 \times 7700 + 3000$$
$$7700 = 2 \times 3000 + 1700$$
$$3000 = 1 \times 1700 + 1300$$
$$1700 = 1 \times 1300 + 400$$
$$1300 = 3 \times 400 + 100$$
$$400 = 4 \times 100 + 0$$

$\mathrm{gcd}(3000,7700) = \mathrm{gcd}(7700,3000) =$
$\mathrm{gcd}(3000,1700) = \mathrm{gcd}(1700,1300) = \mathrm{gcd}(1300,400) =$
$\mathrm{gcd}(400,100) = \mathrm{gcd}(100,0) = 100$

PSN. Using Euclid’s algorithm compute gcd(585,1035)

Recursive version of Euclid GCD

```
function EuclidGCD(a,b)
 Input: a, b (nonnegative integers)
 Output: gcd(a,b)
     if (b == 0)
          return a
     else
          r = a mod b
          return EuclidGCD(b,r)
     endif
 end EuclidGCD
```

Nonrecursive version

```
function EuclidGCD(a,b)
Input: a, b (nonnegative integers)
Output: gcd(a,b)
    while b ≠ 0 do
        Remainder = a mod b
        a = b
        b = Remainder
    endwhile
    return(a)
end EuclidGCD
```

Most iterations performed

Note that if $a < b$, then a and b get swapped after the first iteration, so no need to make this test.

Proposition. Euclid's algorithm in computing gcd(a,b) where $a \geq b$ makes at most 2n iterations where n is the number of binary (base 2) digits of a.

Proof. After one iteration a is replaced with b and b with r and after another iteration b is replaced with r. Thus, after two iterations a is replaced with r. But, the remainder r in binary has at least one fewer digits than a. Thus, after every other iteration a is reduced by at least one binary digit. It follows that the number of iterations performed in computing gcd(a,b) for any a and b where $a \geq b$ is at most twice the number of binary digits of a.

For what input does Euclid's algorithm take the most time?

Answer: a = fib(n), b = fib(n+1), where fib(n) is the nth Fibonacci number.

Fibonacci numbers: 0 1 1 2 3 5 8 13 21 35 56 …

gcd(35,56) = gcd(56,35) = gcd(35,21) = gcd(21,13) = gcd(13,8) = gcd(8,5) = gcd(5,3) = gcd(3,2) = gcd(2,1) = gcd(1,1) = gcd(1,0) = 1

Applications of gcd

- Lowest Common Multiple
- Fraction in Lowest Form
- Cryptographic algorithms such as RSA

Lowest Common Multiple (lcm)

$\mathrm{lcm}(a, b)$ is the smallest multiple of both a and b.

Proposition. $\mathrm{lcm}(a, b) = \frac{ab}{\gcd(a,b)}$.

Example. $a = 585, b = 1035$

$a \times b = 585 \times 1035 = 605475$

$\gcd(585,1035) = 45$

$\mathrm{lcm}\,(585,1035) = 23 \times 585 = 13 \times 1035$

$= 13455$

$= \frac{605475}{45}$

Fraction in Simplest Form

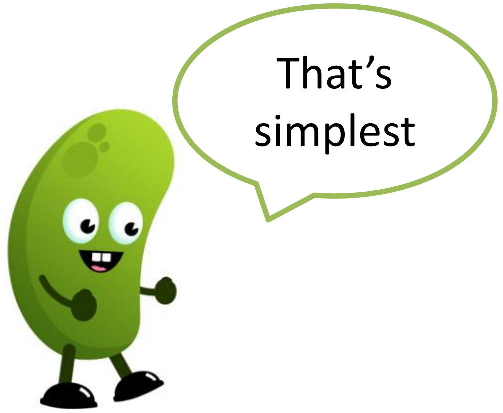
$$\frac{a}{b} \text{ in simplest form is } \frac{a/\gcd(a,b)}{b/\gcd(a,b)}$$

Example. $a = 585, b = 1035$

$$\frac{a}{b} = \frac{585}{1035}$$

$$\frac{a/\gcd(a,b)}{b/\gcd(a,b)} = \frac{585/\gcd(585,1035)}{1035/\gcd(585,1035)}$$

$$= \frac{585/45}{1035/45} = \frac{13}{23}$$



Extended Euclid's algorithm

We now design an extension of Euclid's GCD algorithm that computes integers g, s, t, where $g = \gcd(a,b)$ and

$$g = sa + tb.$$

This algorithm has important applications including use in the design of the **RSA public-key cryptosystem** which is used extensively for encryption and digital signatures on the Internet.

Example of Extended Euclid’s algorithms for $a = 6700, b = 3000$

$6700 = 2 \times 3000 + 700 \Rightarrow 700 = 6700 - 2 \times 3000$

$3000 = 4 \times 700 + 200 \Rightarrow 200 = 3000 - 4 \times 700$

$700 = 3 \times 200 + 100 \Rightarrow 100 = 700 - 3 \times 200$

$200 = 2 \times 100 + 0$

$100 = 700 - 3 \times 200$. Substituting from above we have

$100 = 700 - 3 \times (3000 - 4 \times 700)$. Simplifying we have

$100 = 13 \times 700 - 3 \times 3000$. Substituting from above we have

$100 = 13 \times (6700 - 2 \times 3000) - 3 \times 3000$. Simplifying we have

$100 = 13 \times 6700 - 29 \times 3000$

We have solved $g = \mathrm{gcd}(a, b) = sa + tb$ for $a = 6700, b = 3000$ obtaining $g = 100, s = 13, t = -29$.

PSN. Solve $g = \gcd(a, b) = sa + tb$ for $a = 1035, b = 585$

Extended Euclid GCD

Suppose we have $g = s'b + t'r$

By definition $a = bq + r$, where q is the quotient, so that

$$r = a - bq$$

Substituting this value for r into $g = s'b + t'r$ we obtain

$$g = s'b + t' (a - bq) = t'a + (s' - t'q)b$$

Assigning $s = t'$ and $t = s' - t'q$, we have the desired result

$$g = sa + tb$$

Extended Euclid GCD Algorithm

```
function ExtEuclidGCD(a,b,g,s,t)
     Input: a, b (nonnegative integers)
     Output: return g = gcd(a,b) and integers s and t such that sa + tb = g
        if (b == 0)    //BOOTSTRAP CONDITION
              g = a
              s = 1
              t = 0
        else
              r = a mod b
              q = a/b
              ExtEuclidGCD(b,r,g,s,t)    //recursive call
              stemp = s
              s = t
              t = stemp – t*q
end ExtEuclidGCD
```

Historically Bad Joke

Humphrey Bogart is sitting in his bar in Casablanca, enjoying the sublime beauty of geometry…

He raises his glass and says,

"Here's looking at Euclid."