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

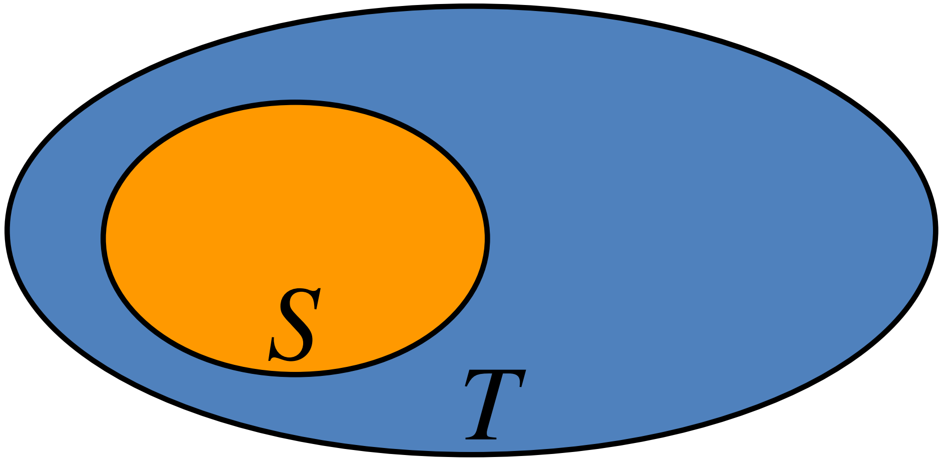
- $S \subseteq T$ ("S is a *subset* of T") means that every element of S is also an element of T.
- Equivalently,

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

- $\varnothing \subseteq S$, $S \subseteq S$.
- $S \supseteq T$ ("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 $S \supset T$.



Venn Diagram equivalent of $S \subset T$

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

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.*,

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

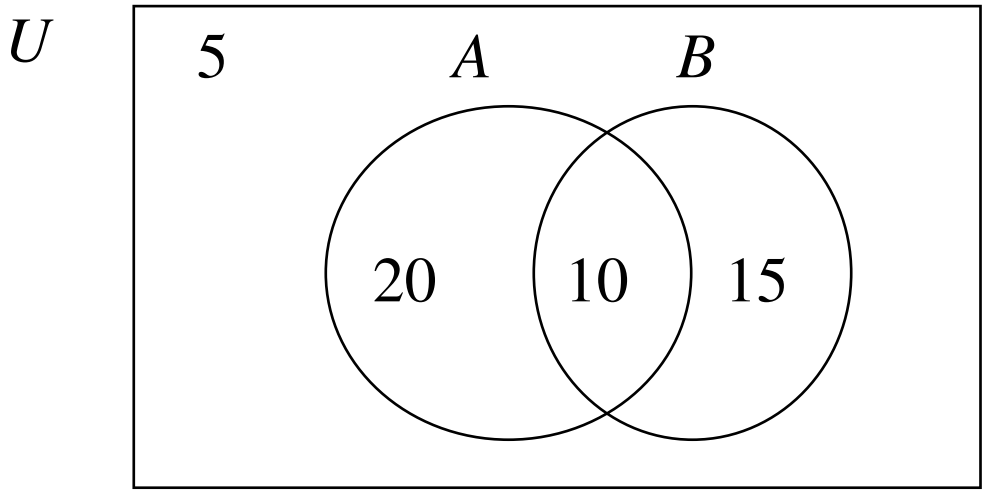
$$|\{\{1,2,3\},\{4,5\}\}| = 2$$

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?

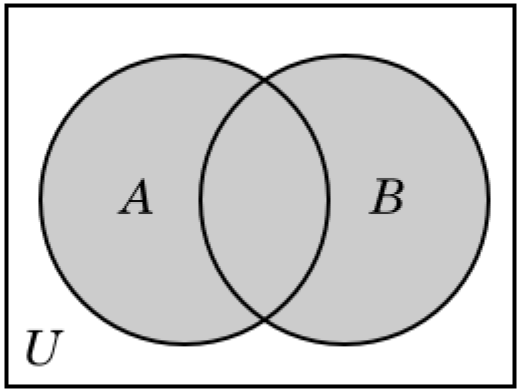


$$|A \cup B| = |A| + |B| - |A \cap B|$$

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

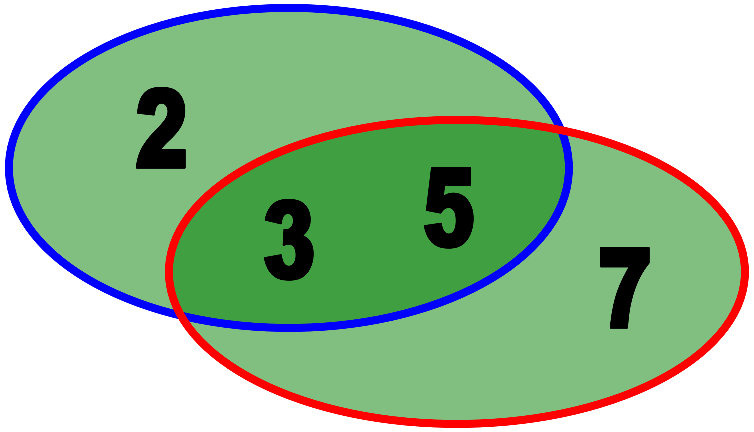
The Union Operator

- 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:



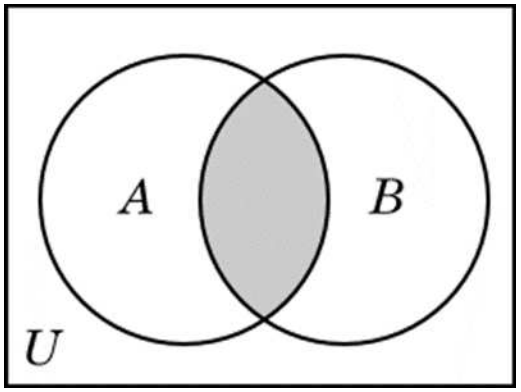
Union Examples

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



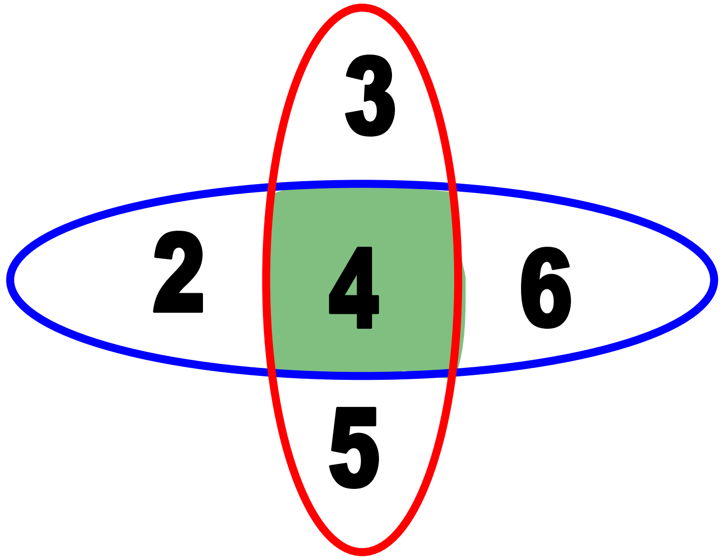
The Intersection Operator

- 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

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



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

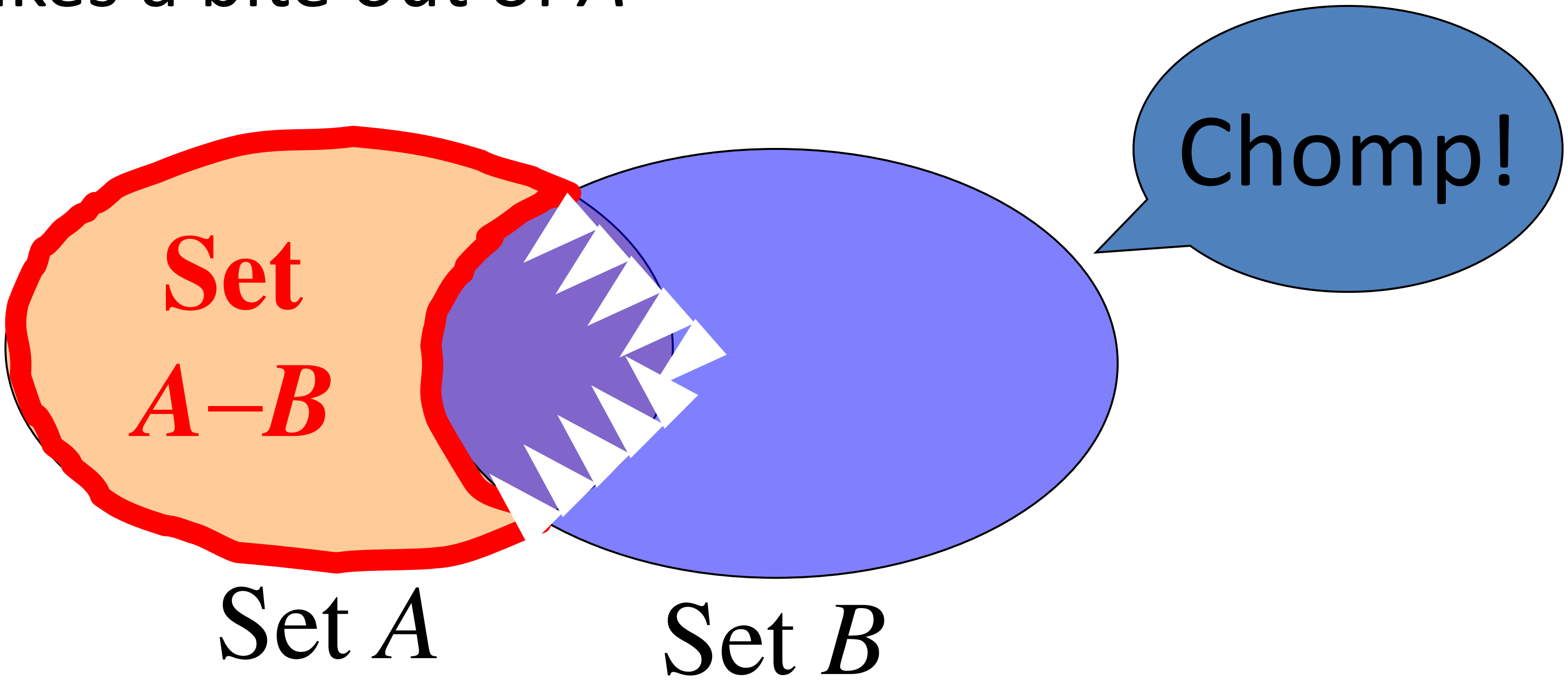
- $\{1,\cancel{2},\cancel{3},4,\cancel{5},6\} - \{2,3,5,7,9,11\} =$

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

- $\mathbf{Z} - \mathbf{N} = \{\ldots, -1, 0, 1, 2, \ldots\} - \{0, 1, \ldots\}$
  $= \{x \mid x \text{ is an integer but not a nat. \#}\}$
  $= \{x \mid x \text{ is a negative integer}\}$
  $= \{\ldots, -3, -2, -1\}$

Set Difference - Venn Diagram

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



Set Complements

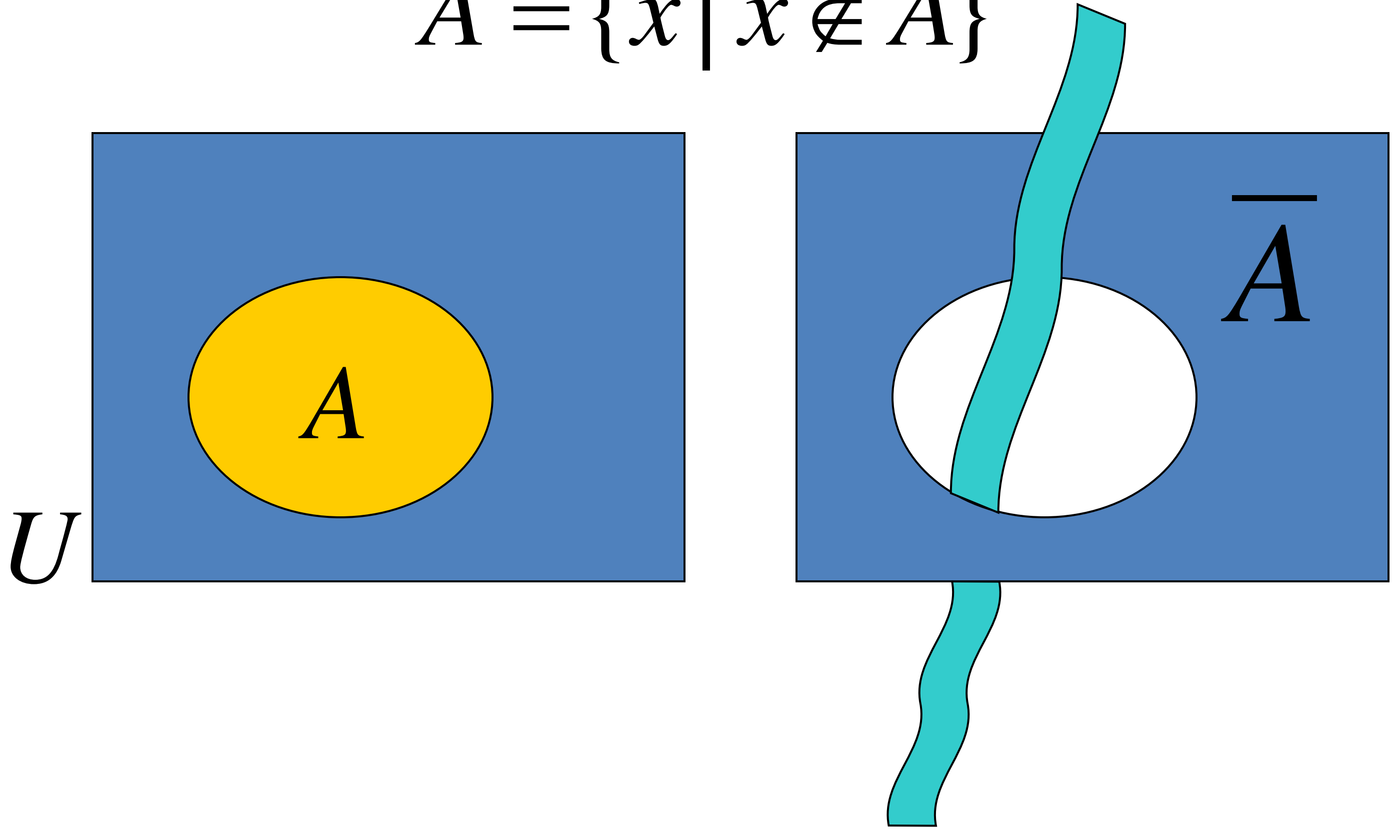
- The *universe of discourse* can itself be considered a set, call it *U*.
- The *complement* of *A*, written $\bar{A}$ or alternatively A^c, is the complement of *A* with respect to *U*, *i.e.*, it is $U - A$.
- *E.g.*, If $U = \{0,1,2,3,4,5,6,7\}$

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

More on Set Complements

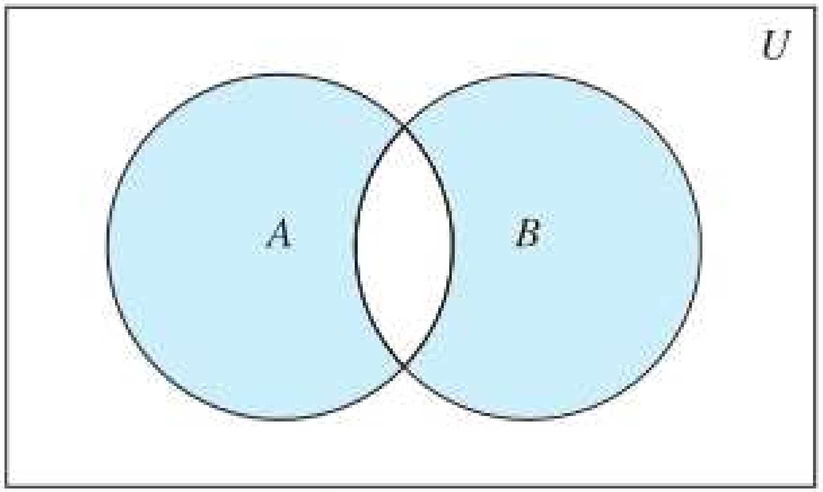
- An equivalent definition, when *U* is clear:

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



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*
- $A \oplus B = A \cup B - A \cap B$



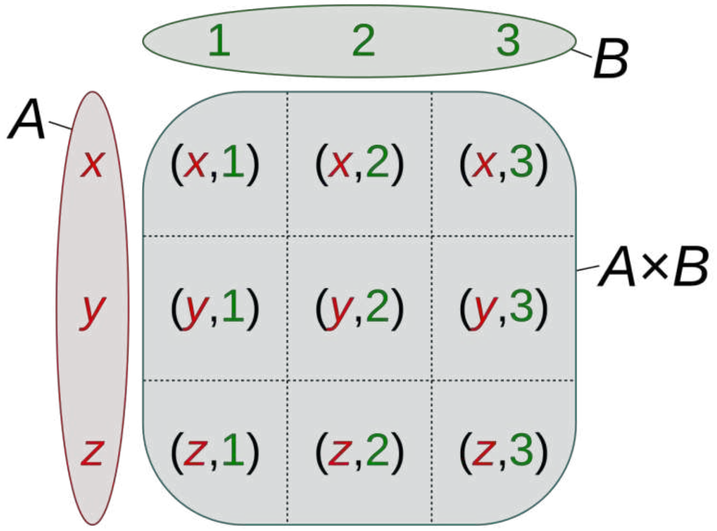
$A \oplus B$ is shaded.

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 ∞

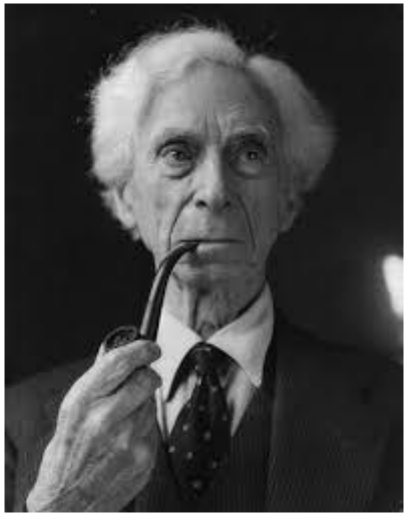
- Conceptually, sets may be *infinite* (*i.e.,* not *finite*, without end, unending).
- Symbols for some special infinite sets:
  N = {0, 1, 2, …} The natural numbers.
  Z = {…, -2, -1, 0, 1, 2, …} The integers.
  R = The “real” numbers, such as
  374.1828471929498181917281943125…
- Is it possible to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

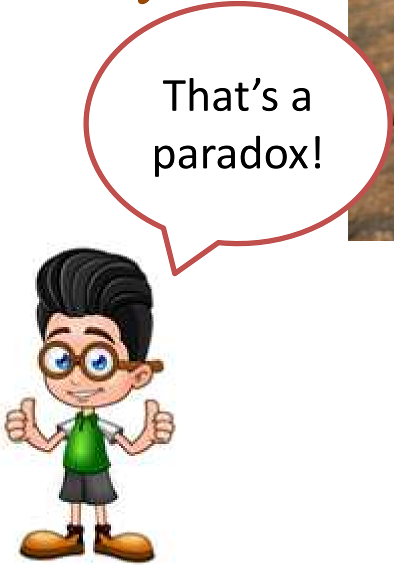
Russell’s Paradox a Milestone in Set Theory

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

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

$P(B) =$

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

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

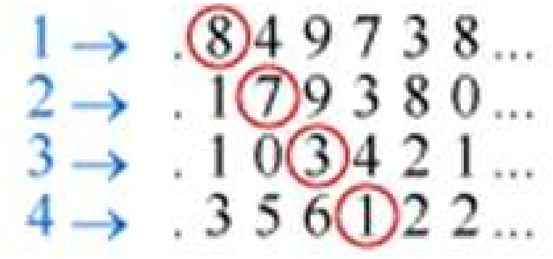
$|\ P(A) \times P(B)| =$

Infinite cardinalities

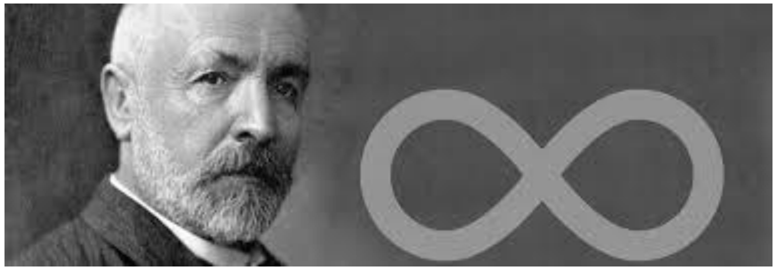
- Cardinality of real numbers is strictly greater than natural numbers.
- In particular, you can't list the set of real numbers between 0 and 1, or equivalently index them with the natural numbers.
- Cantor showed that at least one real number will not be included in the list, so that the subset of real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



Georg Cantor

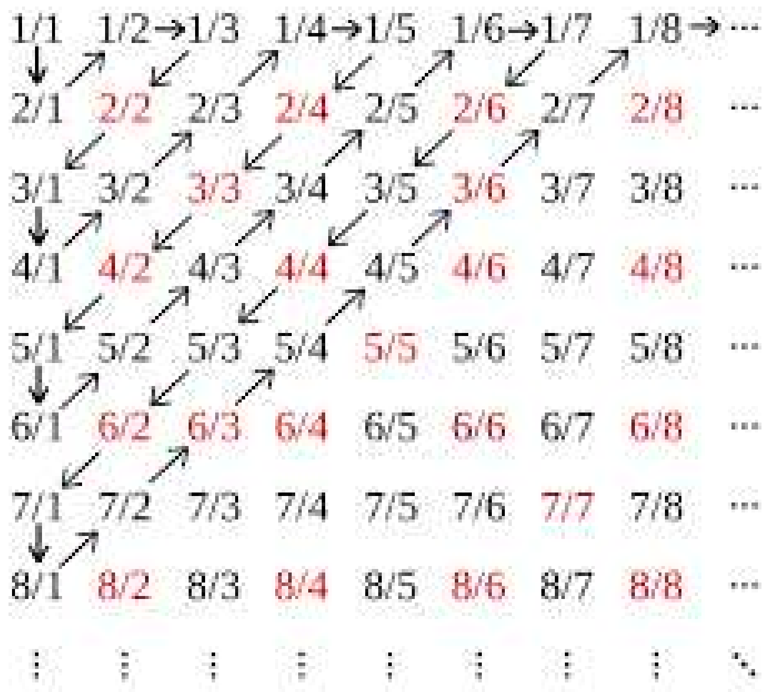


Countable

- 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.

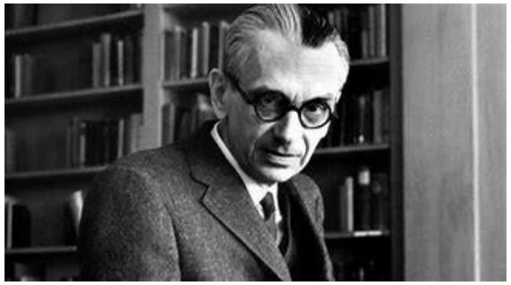
- Despite his efforts Cantor could not resolve CH.

Continuum Hypothesis

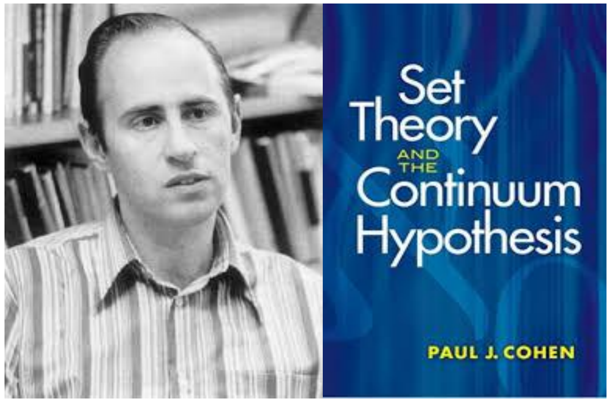
- The problem persisted and was considered so important by Hilbert that he placed it first on his famous list of open problems to be faced by the 20th century. Hilbert also struggled to resolve CH, again without success.

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



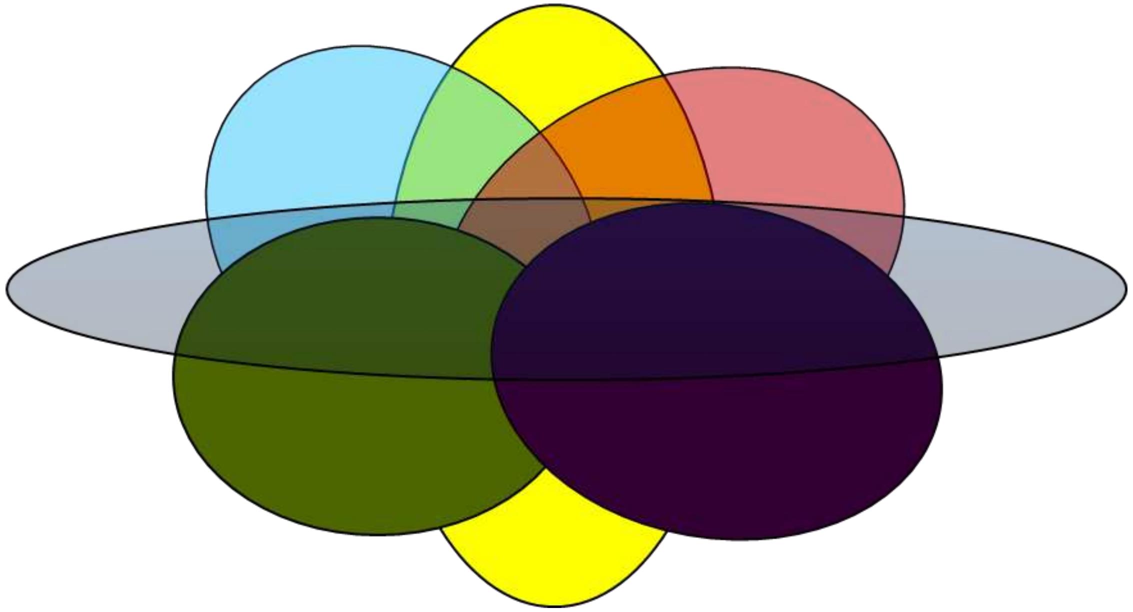
Joke (bad)

Why did the engineering students not finish the lecture video?

They were getting a little ANSI.

ANSI
American National Standards Institute

Principle of Inclusion-Exclusion



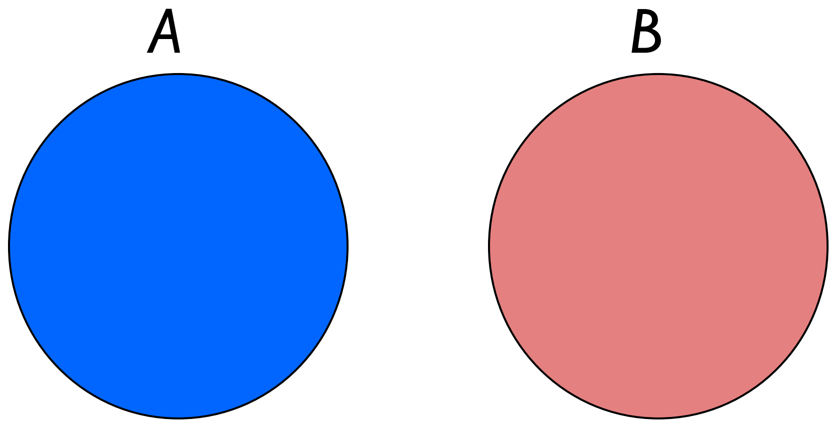
Textbook Reading:

Section 1.5, pp. 34-41

Sum Rule

If sets A and B are disjoint, then

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

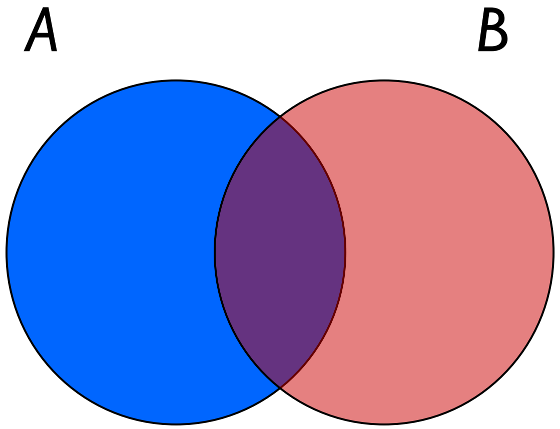


What if A and B are not disjoint?

Inclusion-Exclusion (2 sets)

For two arbitrary sets *A* and *B*

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



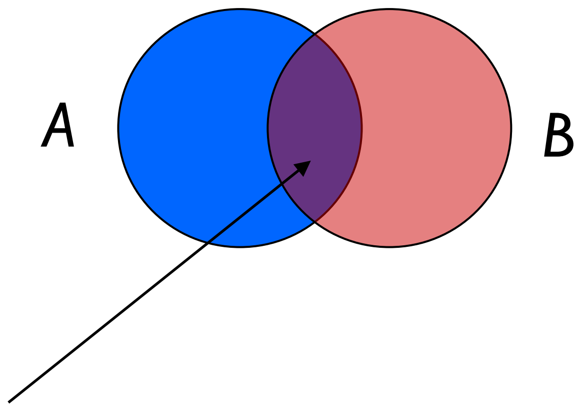
Example Inclusion-Exclusion (2 sets)
How many numbers from 1 to 1000 are multiples of 3 or 5

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

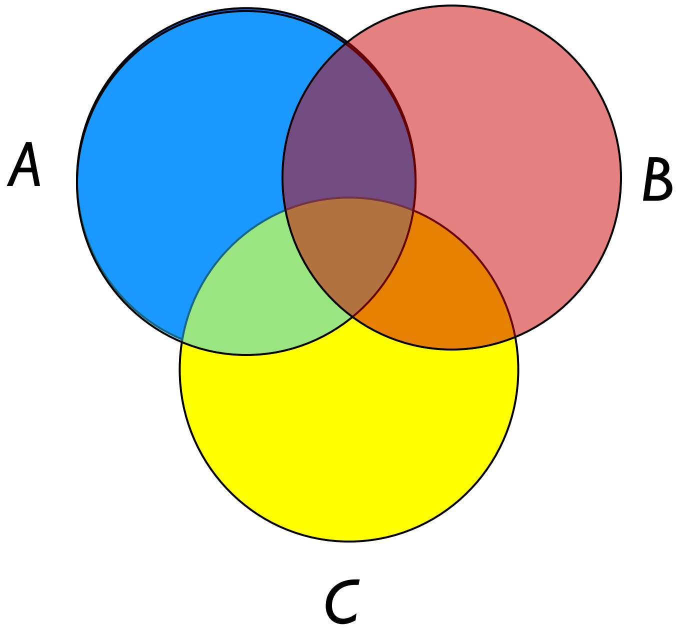
So, by the inclusion-exclusion principle, we have |S| = |A| + |B| – |A ∩ B| = 467.

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.

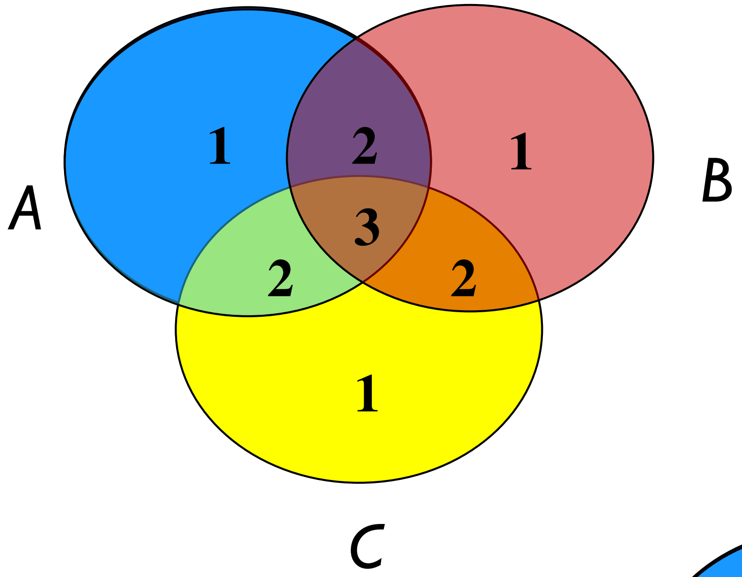
Inclusion-Exclusion (3 sets)



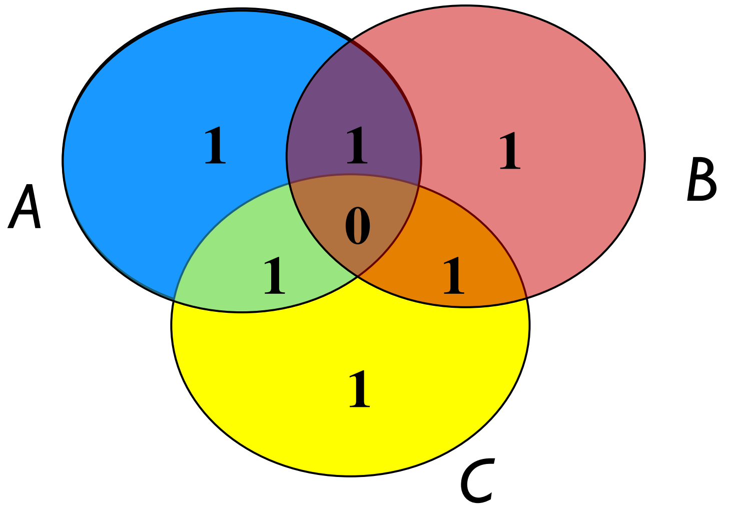
$$\begin{aligned}|A \cup B \cup C| = {} & |A| + |B| + |C| \\ & - |A \cap B| - |A \cap C| - |B \cap C| \\ & + |A \cap B \cap C|\end{aligned}$$

Proof Inclusion-Exclusion (3 sets)

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



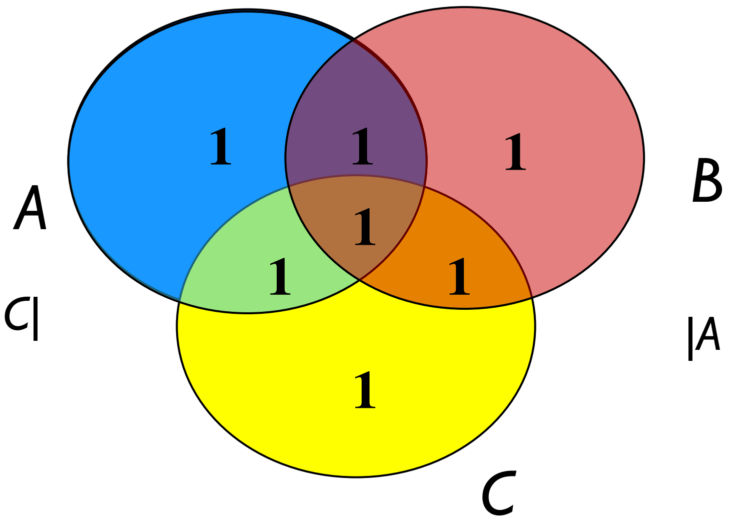
$|A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C|$



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

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

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



$|A \cup B \cup C|$

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.

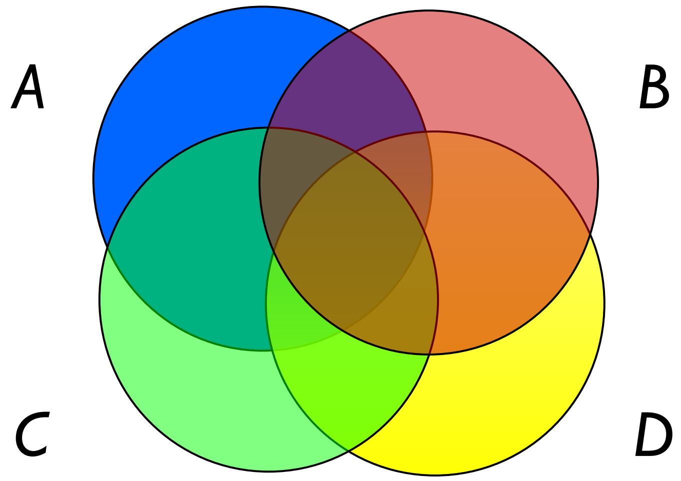
$$\boxed{|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|}$$

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

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

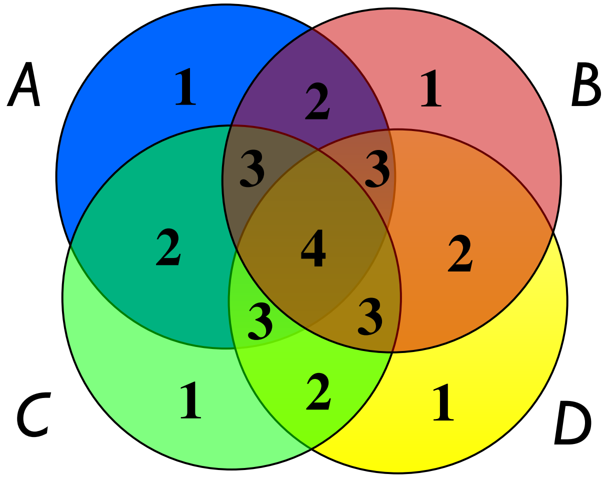
PSN. Using set complement and the Inclusion-Exclusion with 3 sets, obtain a formula for the number of numbers between 1 and 500, inclusive, that are relatively prime to 60.

Inclusion-Exclusion (4 sets)

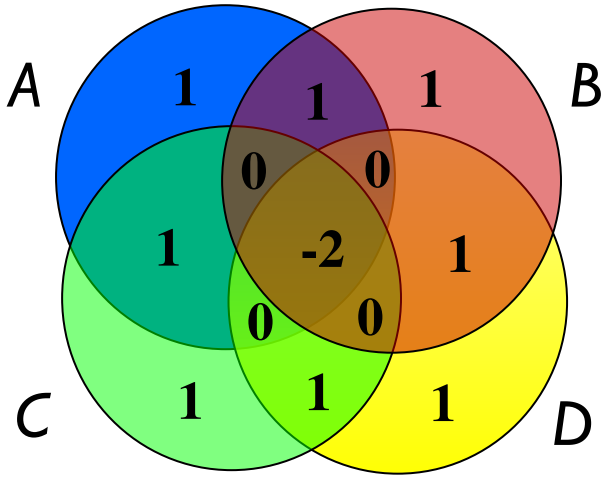


$$
\begin{aligned}
|A \cup B \cup C \cup D| = {} & |A| + |B| + |C| + |D| \\
& - |A \cap B| - |A \cap C| - |A \cap D| - |B \cap C| - |B \cap D| - |C \cap D| \\
& + |A \cap B \cap C| + |A \cap B \cap D| + |A \cap C \cap D| + |B \cap C \cap D| \\
& - |A \cap B \cap C \cap D|
\end{aligned}
$$

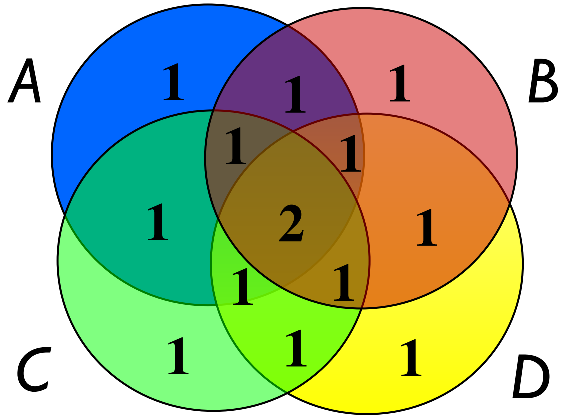
$|A| + |B| + |C| + |D|$



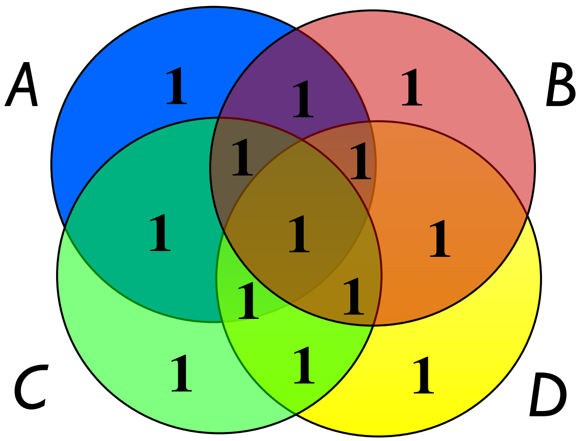
$|A| + |B| + |C| + |D|$
$- |A \cap B| - |A \cap C| - |A \cap D| - |B \cap C| - |B \cap D| - |C \cap D|$



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

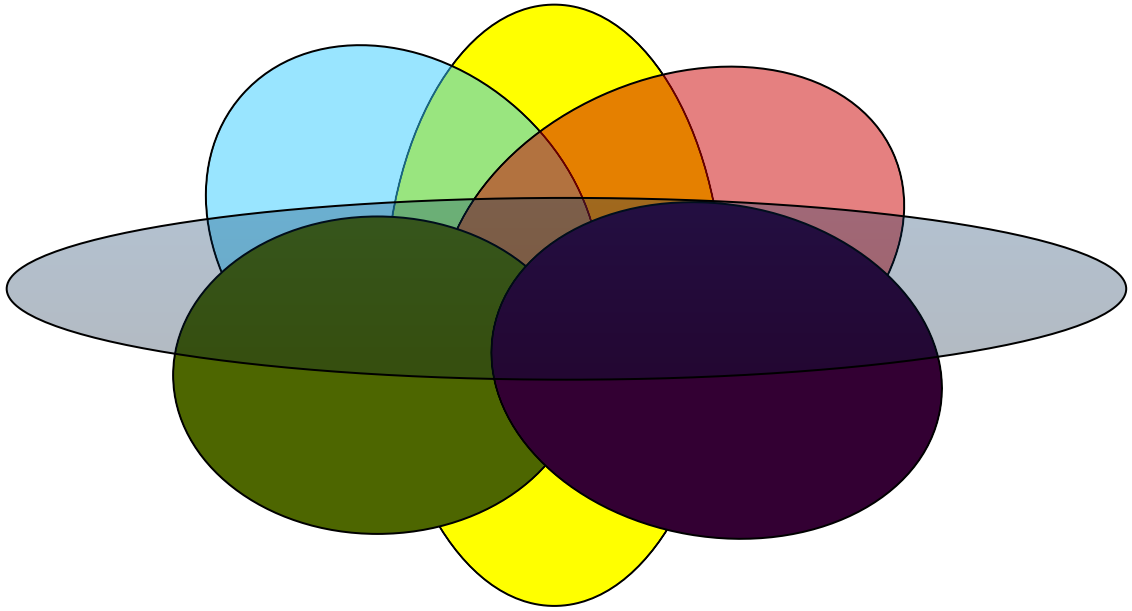


$|A| + |B| + |C| + |D|$
$- |A \cap B| - |A \cap C| - |A \cap D| - |B \cap C| - |B \cap D| - |C \cap D|$
$+ |A \cap B \cap C| + |A \cap B \cap D| + |A \cap C \cap D| + |B \cap C \cap D|$
$- |A \cap B \cap C \cap D| = |A \cup B \cup C \cup D|$



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

…

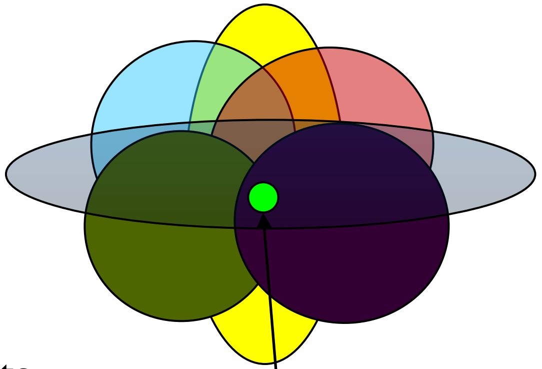
+ $(-1)^{n+1} \times$ sum of sizes of intersections of all n sets

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

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 $P(1), P(2), . . . , P(n), . . .$ 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

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

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**.

$1 + 3 + 5 + \ldots + 2k - 1 + 2k + 1$
$= (1 + 3 + 5 + \ldots + 2k - 1) + 2k + 1$
$= k^2 + 2k + 1$ (by **Induction Hypothesis**)
$= (k + 1)^2$

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.

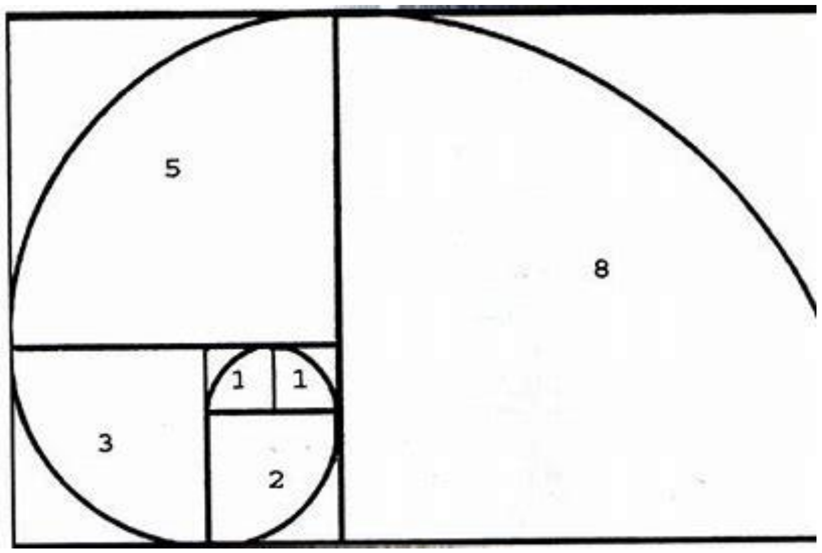
Fibonacci Numbers

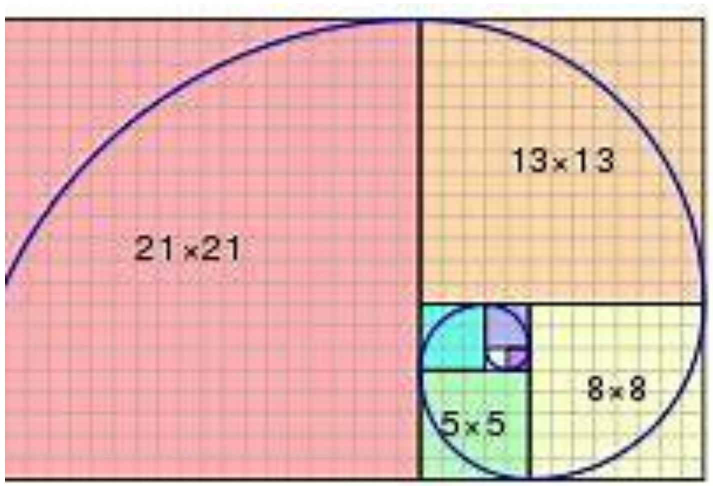
The n^{th} Fibonacci number $Fib(n)$ is defined by the recurrence relations

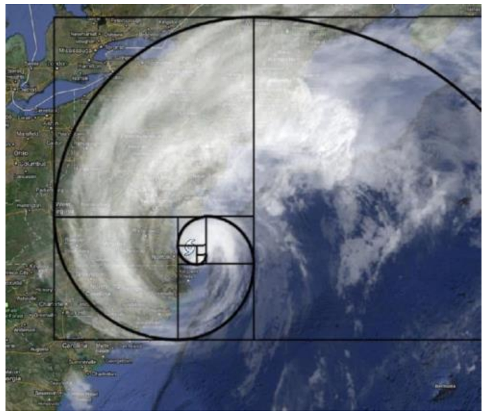
$Fib(n) = Fib(n-1) + Fib(n-2)$, $n \geq 2$, $Fib(0) = 0$, $Fib(1) = 1$.

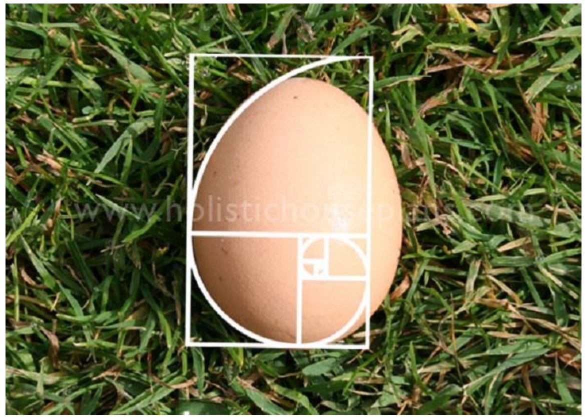
This generates sequence: 0 1 1 2 3 5 8 13 21 34 …

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

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

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 $Fib(n)$:

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

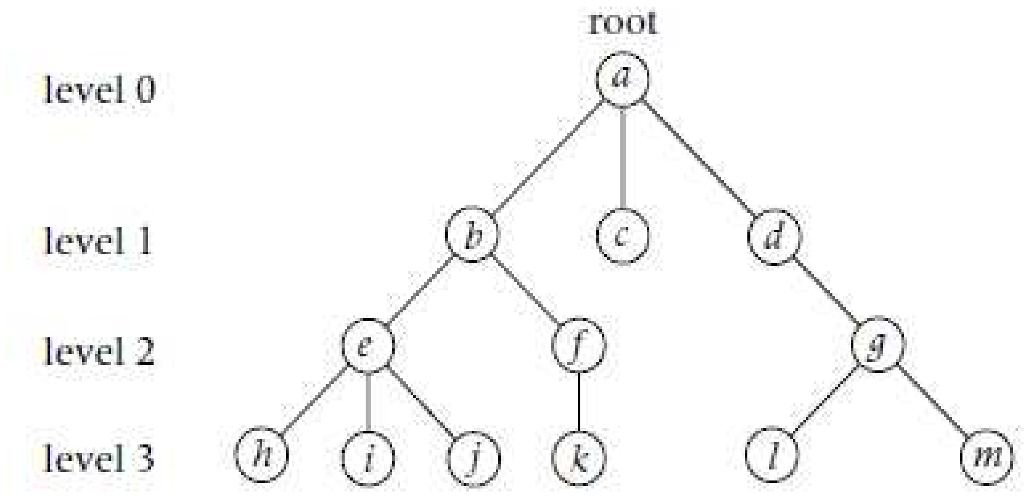
Trees

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)

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

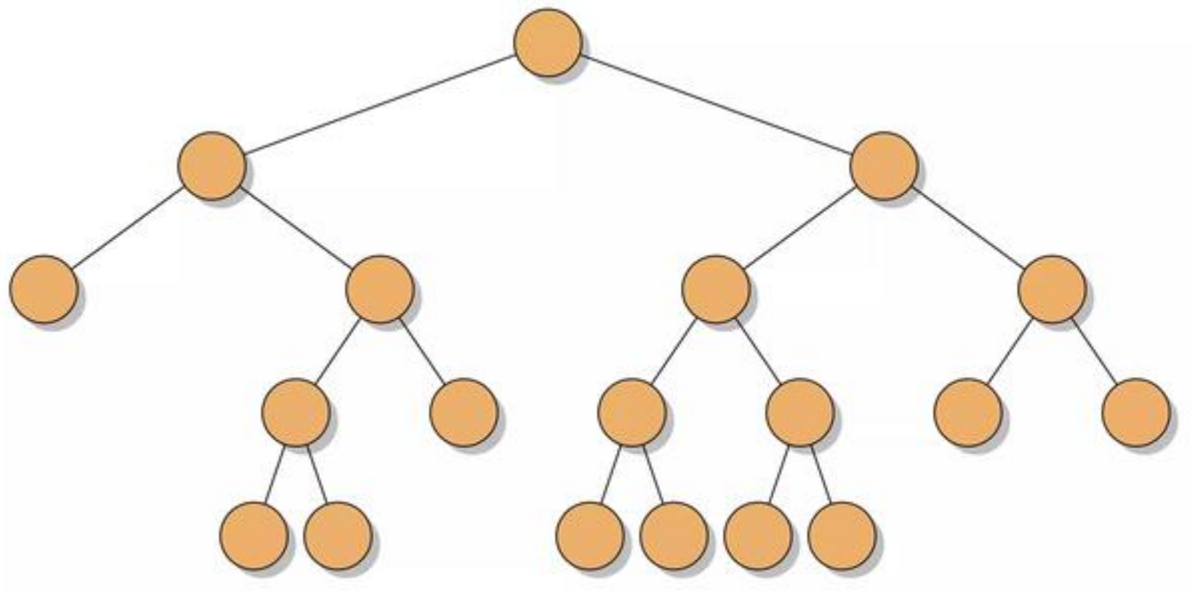
Thus, we have

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

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.,

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

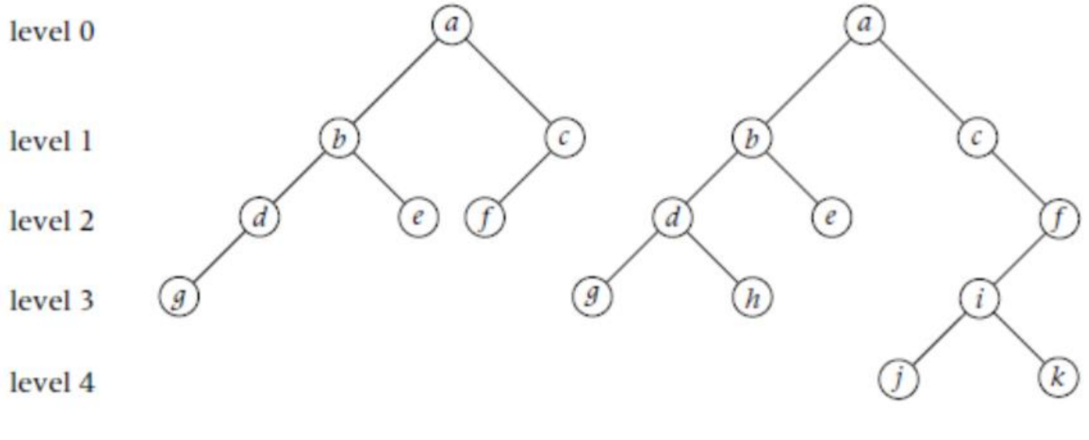
Thus,

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

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

Binary Trees

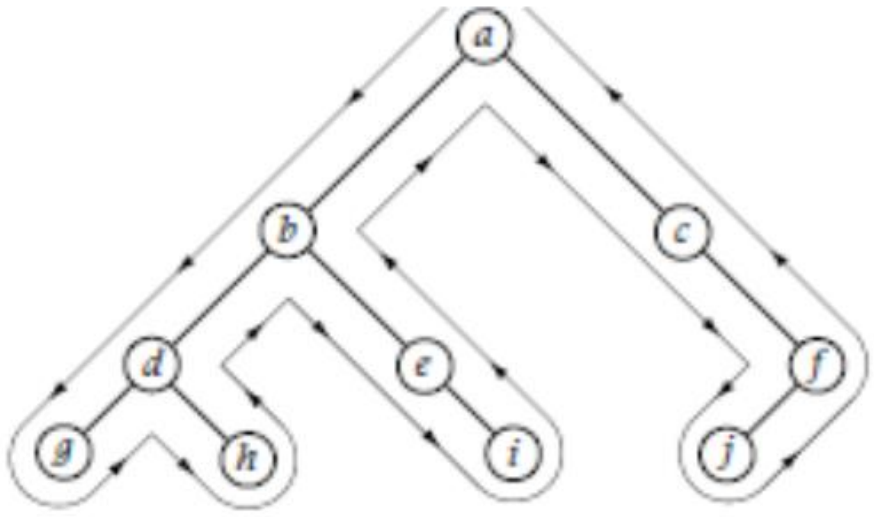
A binary tree is a tree where every node has **at most** 2 children and we identify children as either a left child or right child.



(a) (b)

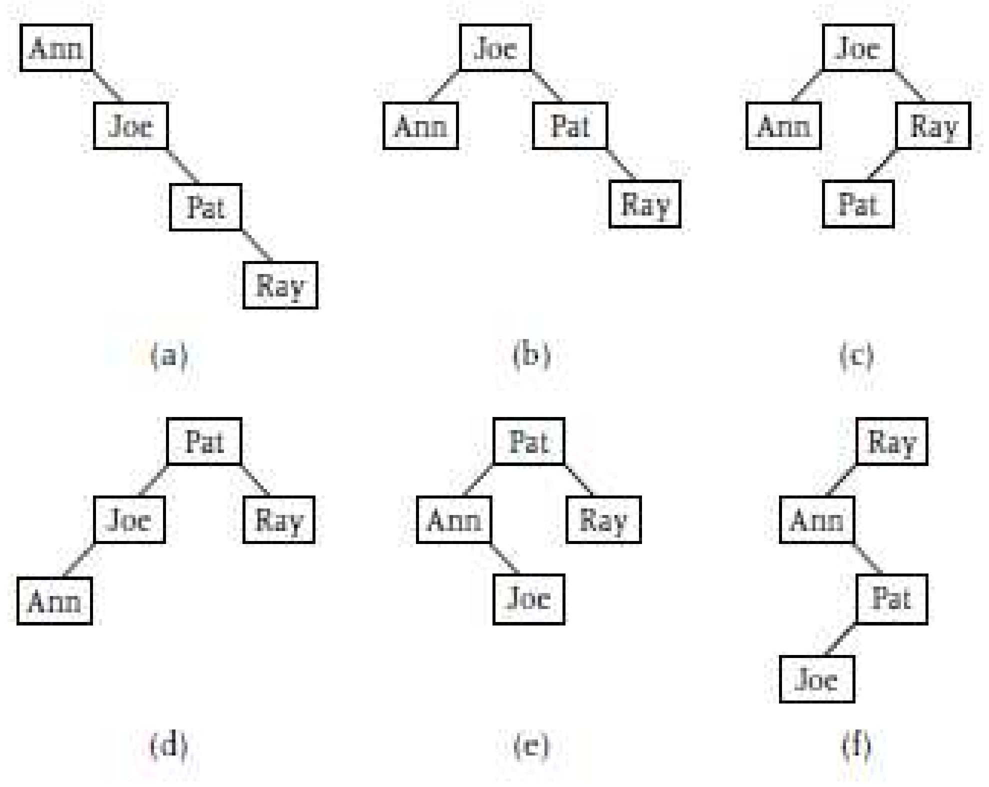
Inorder Traversal

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

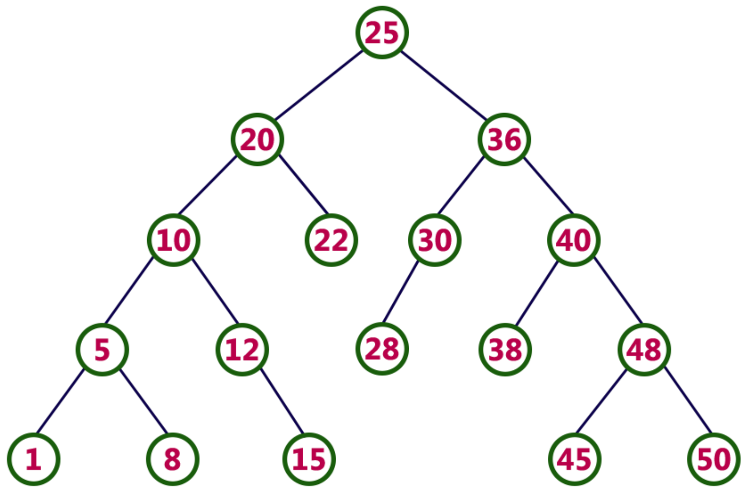


Binary Search Trees

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.

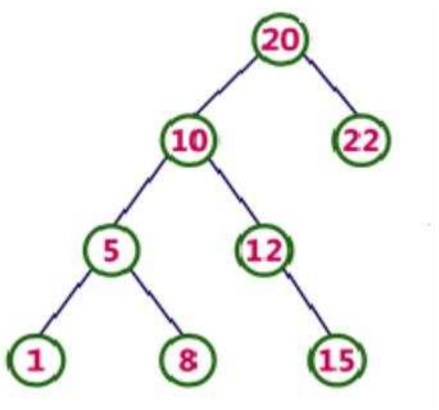
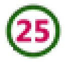
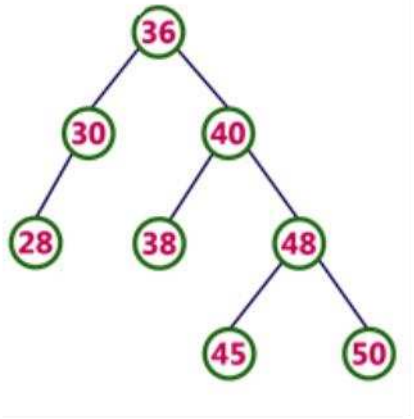
Applying Induction Hypothesis

- 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*

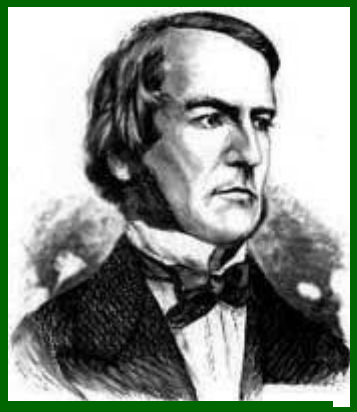
Foundations of Logic

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	¬
Conjunction operator	AND	Binary	∧
Disjunction operator	OR	Binary	∨
Exclusive-OR operator	XOR	Binary	⊕
Implication operator	IMPLIES	Binary	→
Biconditional operator	IFF	Binary	↔

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* "$\land$" (*AND*) combines two propositions to form their logical *conjunction*.

$\land$ND

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

Remember: "$\land$" points up like an "A", and it means "$\land$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.

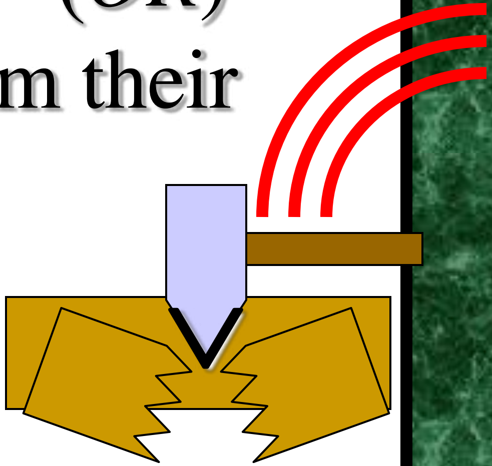
Operand columns

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* "∨" (*OR*) combines two propositions to form their logical *disjunction*.



p="My car has a bad engine."

q="My car has a bad carburetor."

p∨*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 "∨" 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." = $f \wedge (g \vee s)$
  - $(f \wedge g) \vee s$ would mean something different
  - $f \wedge g \vee s$ would be ambiguous
- By convention, "¬" takes *precedence* over both "∧" and "∨".
  - $\neg s \wedge f$ means $(\neg s) \wedge f$, **not** $\neg (s \wedge f)$

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."

$r \land \neg p$ = "The lawn was wet this morning, and it didn't rain last night."

$\neg r \lor p \lor q$ = "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 English "or" can be ambiguous 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 inclusive.

The *Implication* Operator

The *implication* $\underbrace{p}_{\text{antecedent}} \rightarrow \underbrace{q}_{\text{consequent}}$ 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

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

p	q	$p \rightarrow q$
F	F	T
F	T	T
T	F	**F** } The only False case!
T	T	T

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 $p \to q$

- "*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	q	$p \leftrightarrow q$
F	F	T
F	T	F
T	F	F
T	T	T

- $p \leftrightarrow q$ 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.

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

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

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

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...

$(\neg p \lor q) \lor ((p \lor r) \land \neg(p \land r))$ [∨ commutes]

$\Leftrightarrow (q \lor \neg p) \lor ((p \lor r) \land \neg(p \land r))$ [∨ associative]

$\Leftrightarrow q \lor (\neg p \lor ((p \lor r) \land \neg(p \land r)))$ [distrib. ∨ over ∧]

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

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

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

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

[identity] $\Leftrightarrow q \lor (\neg p \lor \neg(p \land r)) \Leftrightarrow$ *cont.*

End of Long Example

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

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

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

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

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

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

Q.E.D. (quod erat demonstrandum)

(Which was to be shown.)

PSN. Give alternate proof using truth tables that

$$(p \wedge \neg q) \rightarrow (p \oplus r) \Leftrightarrow \neg p \vee q \vee \neg r.$$

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. $p \vee \neg p$

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

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 $I(\varphi)$ = T for every interpretation I.

φ is a contradiction iff $I(\varphi)$ = 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	$\neg p \wedge \neg q \wedge \neg r$
F F T	F	**T**	$\neg p \wedge \neg q \wedge r$
F T F	T	F	$\neg p \wedge q \wedge \neg r$
F T T	T	F	$\neg p \wedge q \wedge r$
T F F	F	**T**	$p \wedge \neg q \wedge \neg r$
T F T	F	**T**	$p \wedge \neg q \wedge r$
T T F	T	F	$p \wedge q \wedge \neg r$
T T T	T	F	$p \wedge q \wedge r$

First put negation of formula in DNF:

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

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:

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

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

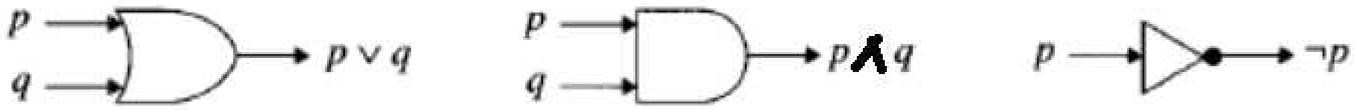
- **Expression Trees**, page 94 of text
- **Combinatorial Circuits**, using gates to represent formulas, page 98

Expression Trees

PSN. Give the expression tree for the formula

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

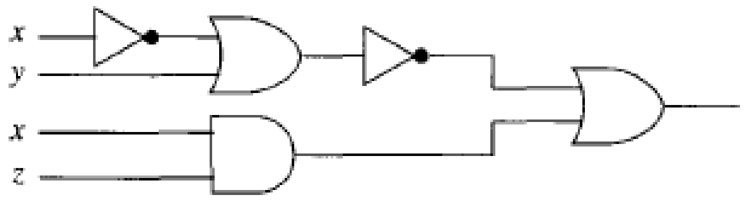
Gates to represent formulas



OR, AND and NOT Gates

Combinatorial Circuit

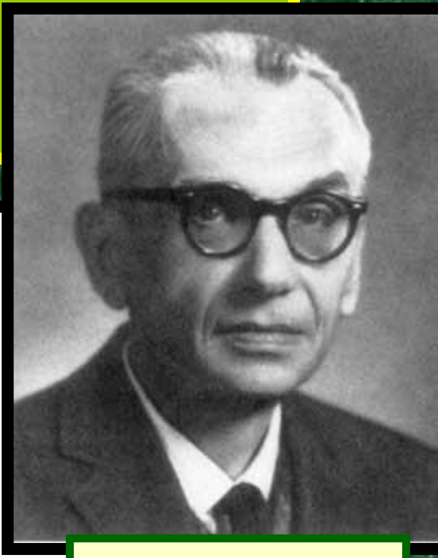
A set of gates is called a **combinatorial circuit** or **combinatorial network.**



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.
- "∀" is the FOR∀LL or *universal* quantifier. $\forall x\, P(x)$ means *for all* x in the u.d., P holds.
- "∃" is the ∃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* $P(x)$, $\exists x\, P(x)$, 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:

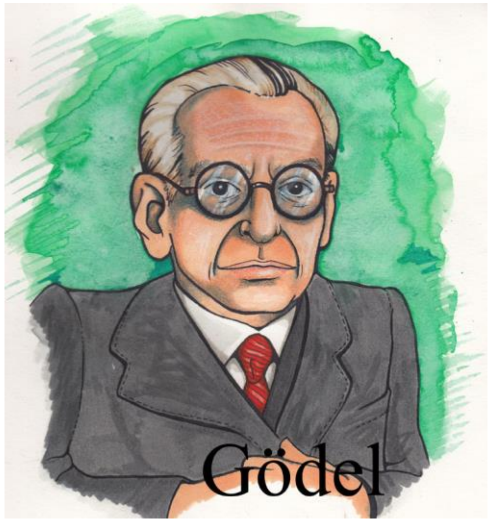
$$\left(\lim_{x \to a} f(x) = L\right) \Leftrightarrow$$

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

Good Ole Days

No one uses logic anymore.

I miss the Godel days.



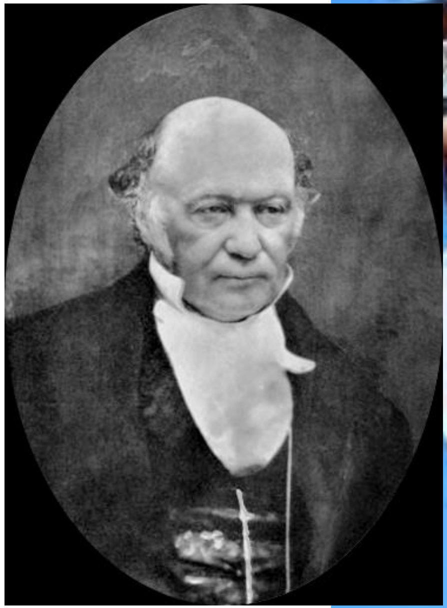
NP-Completeness

Textbook reading:

Section 2.5.6, Page 129

Hamiltonian Cycle

Sir William Rowen Hamilton’s Icosian Game



Goal of Icosian Game

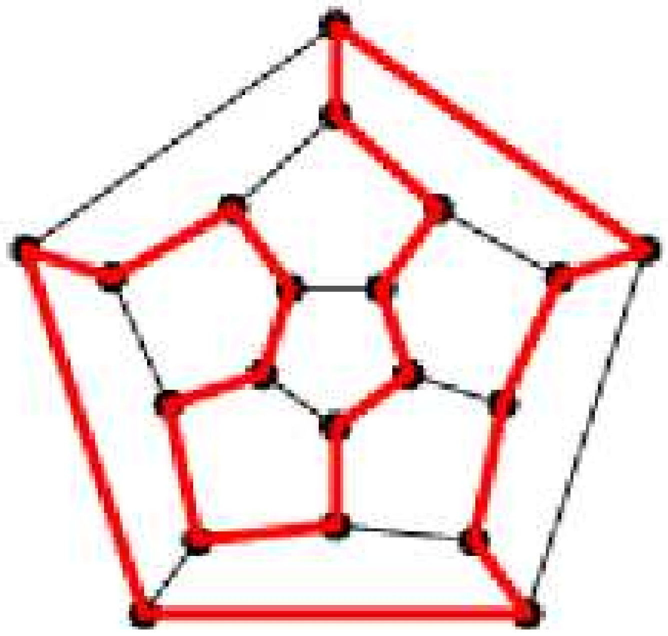
Vertices of the icosahedron represent cities. The goal is to perform a tour of the 20 cities and return to the starting vertex, following an edge of the icosahedron to move between two cities.

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?

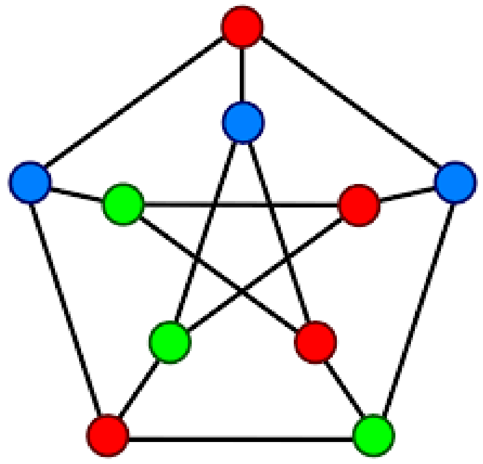
Solution to Icosian Game

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 $A = \{a_0, a_1, . . . , a_{n-1}\}$, 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.

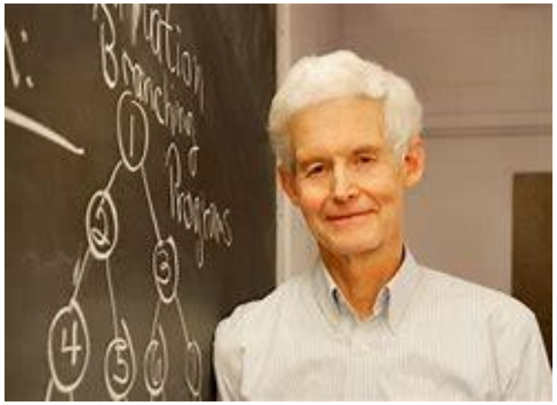
NP-Complete Problems Exist!

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

$y_1, \ldots, y_j \in \{ x_1, \ldots, x_n\} \cup \{\bar{x}_1, \ldots, \bar{x}_n\}$ 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 $y_1, \ldots, y_j$ 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 $y_1, \ldots, y_j$ 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

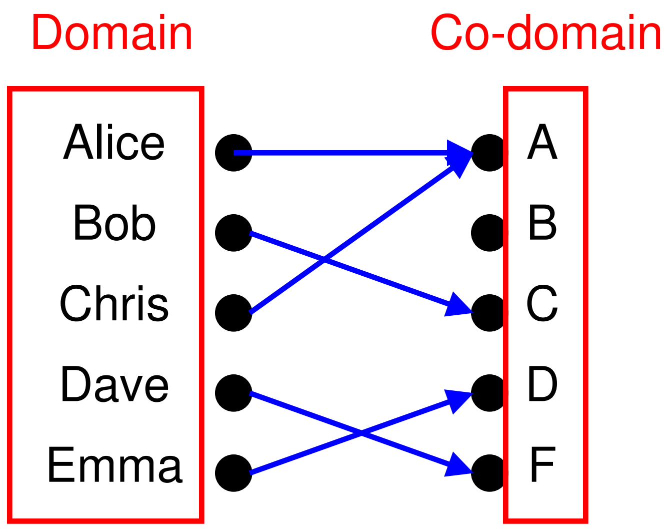
Section 4.6, pp. 253-257 (Pigeonhole Principle)

Definition of a function

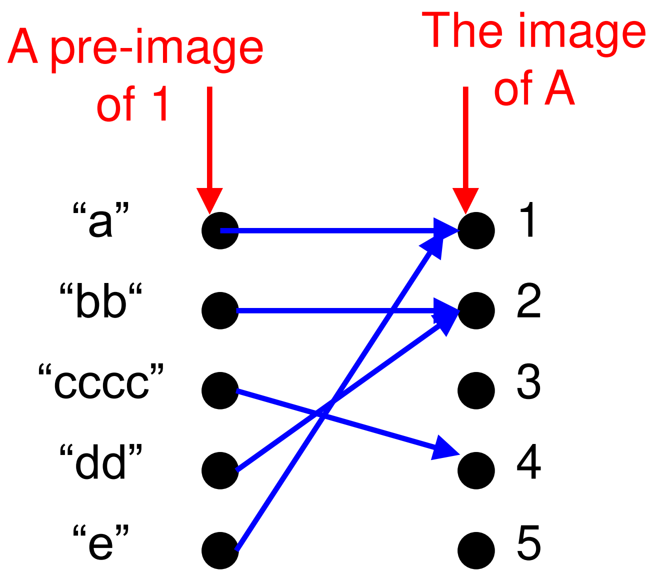
- A function is a binary relation between two sets A (domain) and B (co-domain), i.e., a subset of A×B, so that each element in A occurs in exactly one pair.

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

Examples of functions

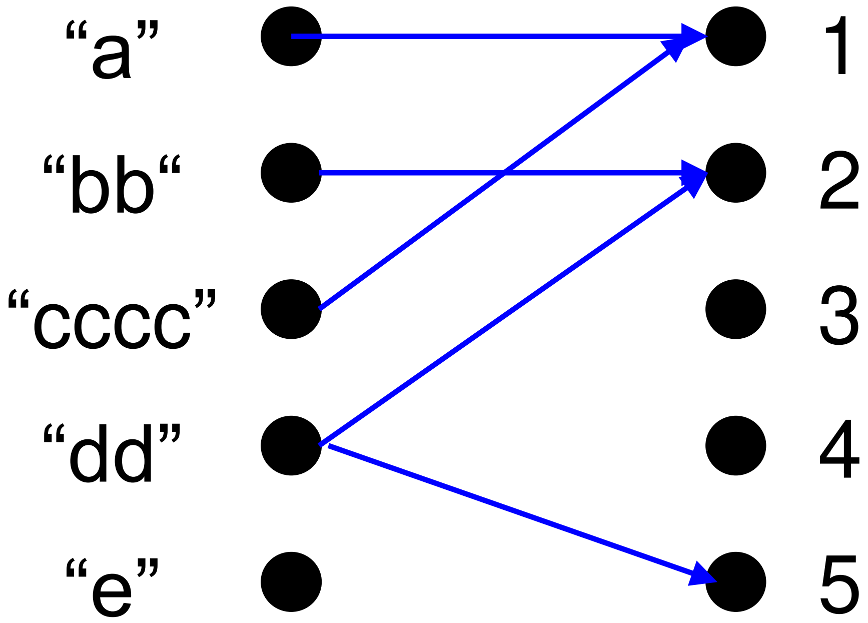


A class grade function

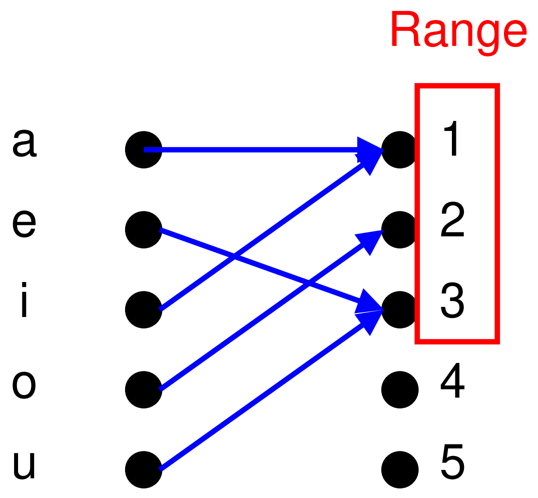


A string length function

Not a function



Range



Some function…

Function arithmetic

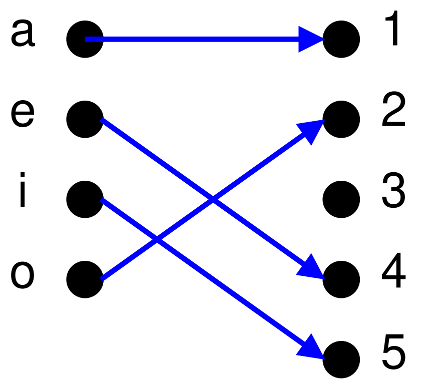
- Let $f_1(x) = 2x$
- Let $f_2(x) = x^2$

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

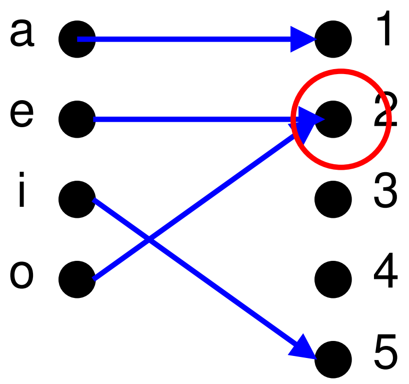
- $f_1*f_2 = (f_1*f_2)(x) = f_1(x)*f_2(x) = 2x*x^2 = 2x^3$

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

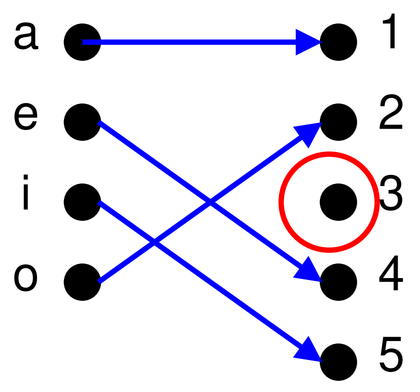


A function that is not one-to-one

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

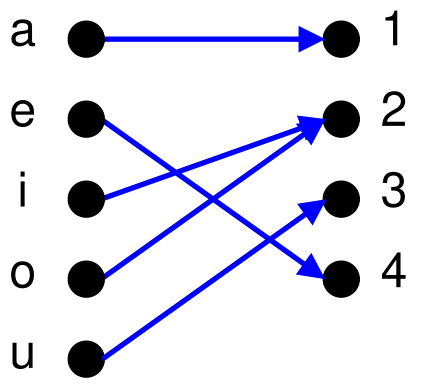
- Note that there can be un-used elements in the co-domain



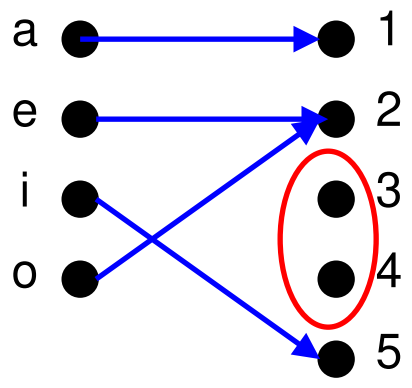
A one-to-one function

Onto functions

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



An onto function

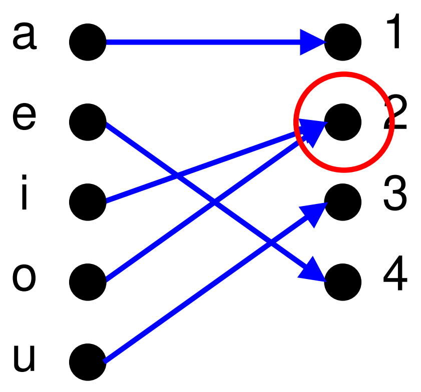


A function that is not onto

More on onto

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

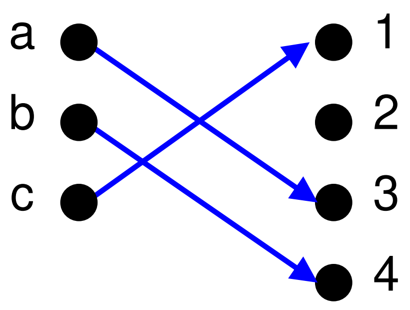
- Note that there can be multiply used elements in the co-domain



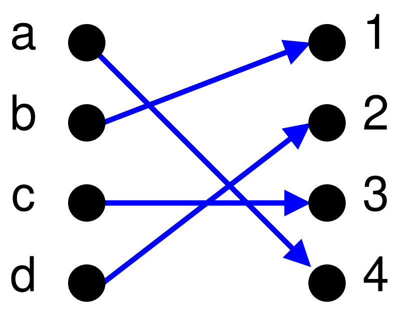
An onto function

Onto vs. one-to-one

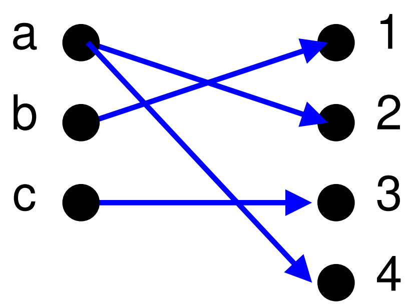
- Are the following functions onto, one-to-one, both, or neither?



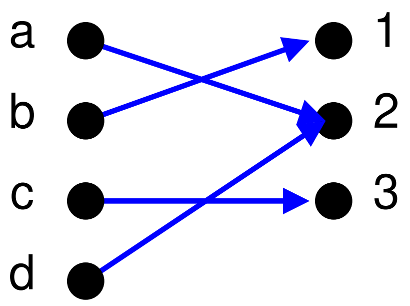
1-to-1, not onto



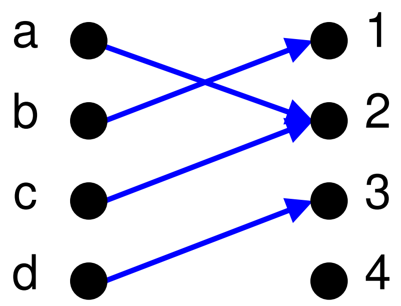
Both 1-to-1 and onto



Not a valid function



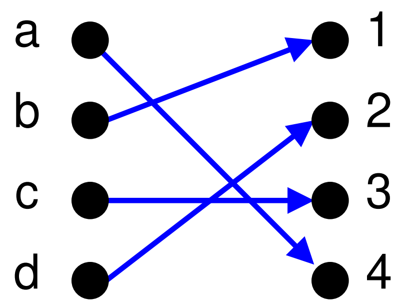
Onto, not 1-to-1



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

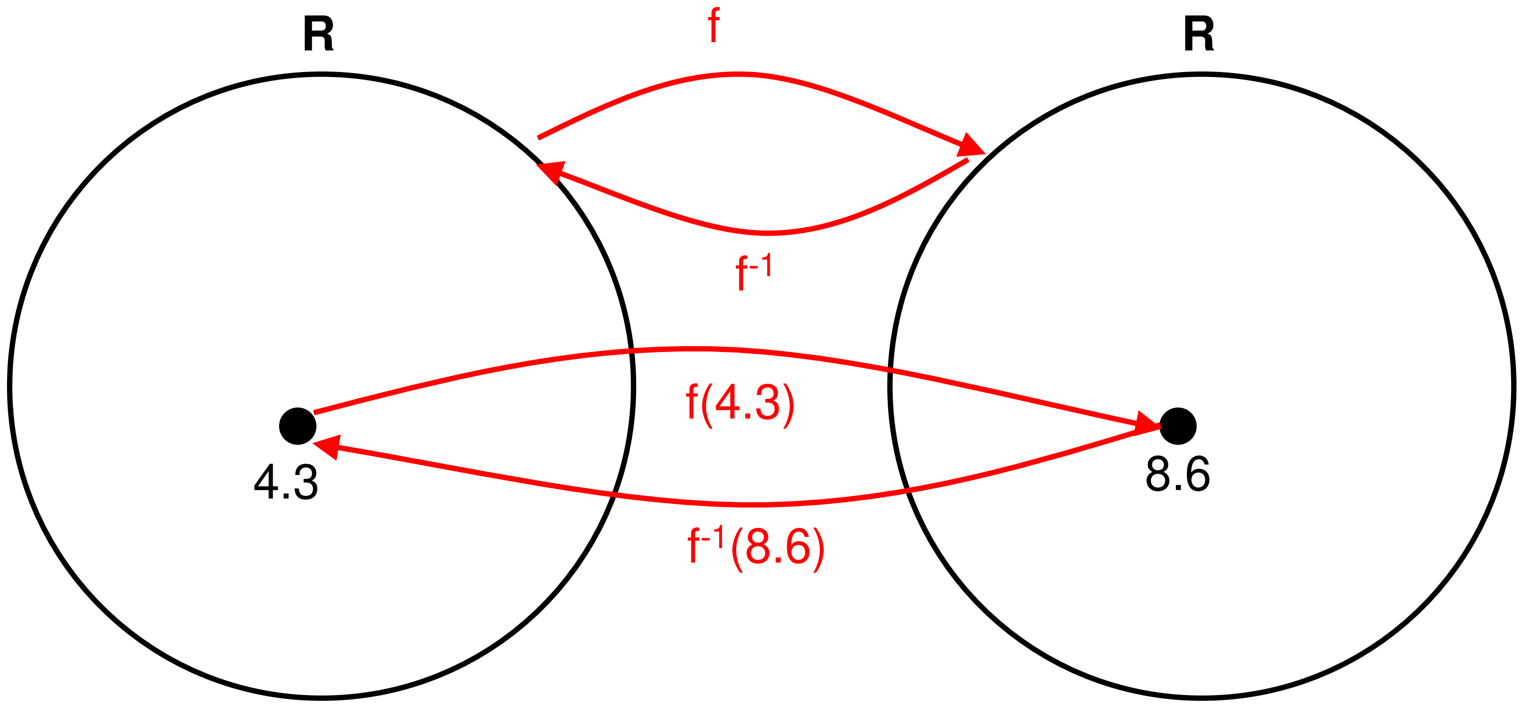
- A function such that the image and the pre-image are ALWAYS equal

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

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

Inverse functions

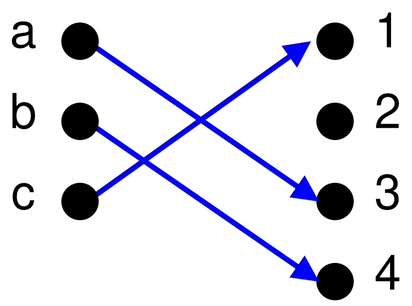
Let f(x) = 2*x



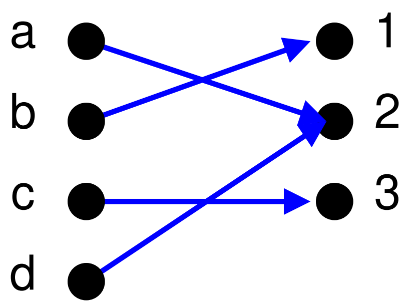
Then $f^{-1}(x) = x/2$

More on inverse functions

- Can we define the inverse of the following functions?



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



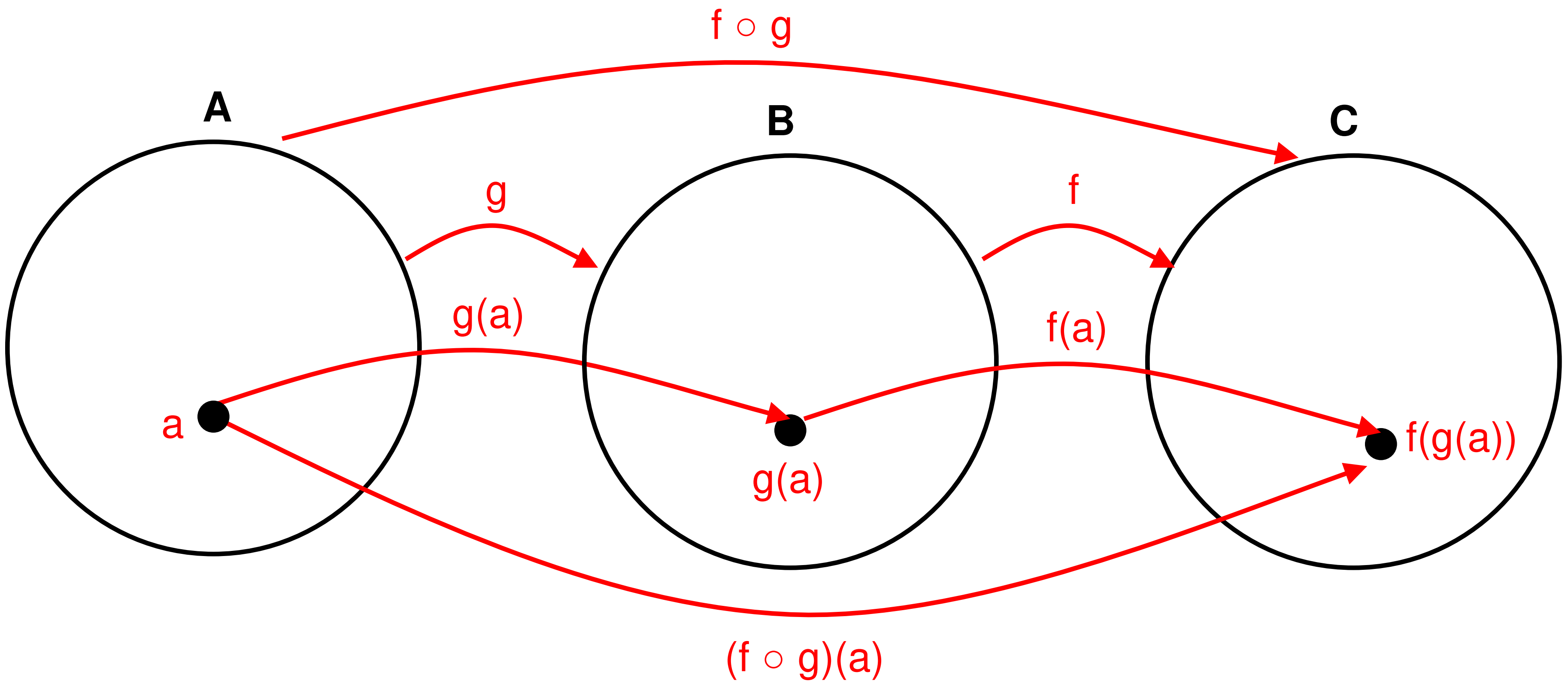
What is $f^{-1}(2)$?
Not 1-to-1!

- An inverse function is ONLY defined on a bijection

Compositions of functions

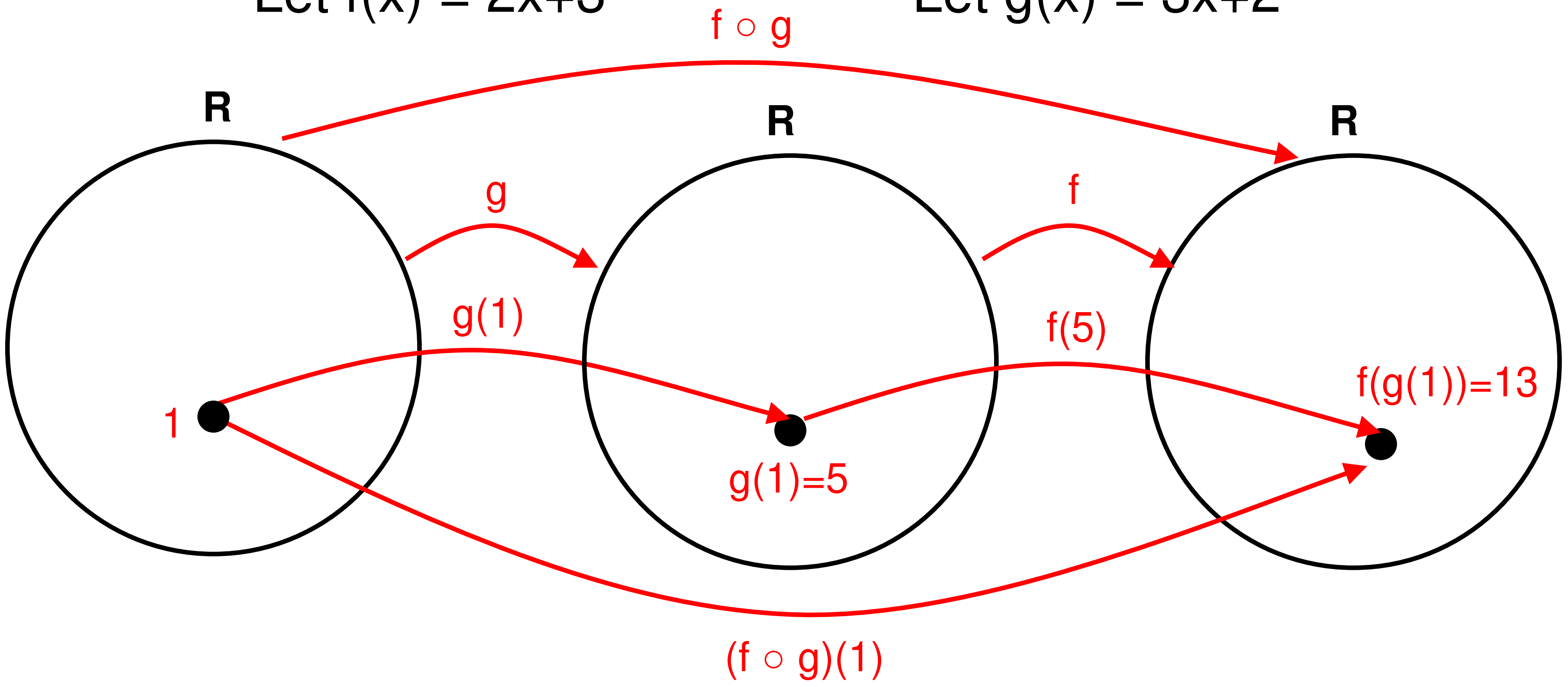
- Let $(f \circ g)(x) = f(g(x))$
- Let $f(x) = 2x+3$ Let $g(x) = 3x+2$
- $g(1) = 5$, $f(5) = 13$
- Thus, $(f \circ g)(1) = f(g(1)) = 13$

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,

$$\mathrm{P} \geq \lceil N/k \rceil .$$

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)?

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

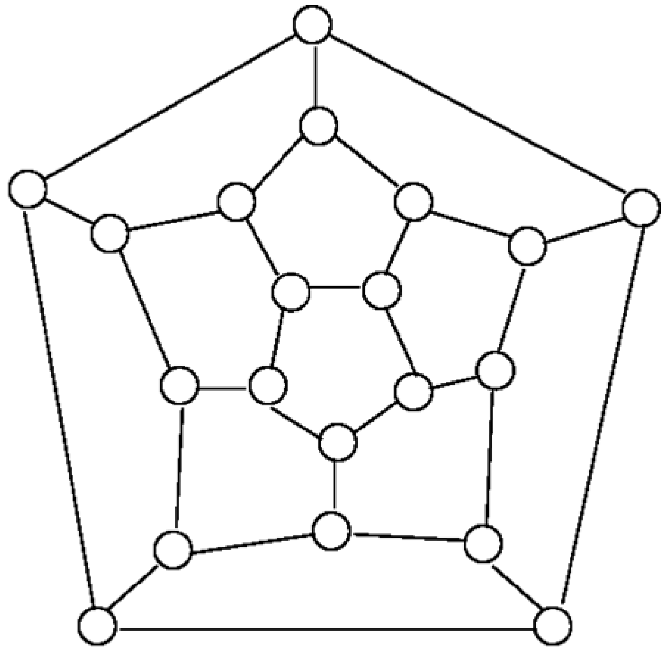
Binary Relations on a Set, cont'd

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

- 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

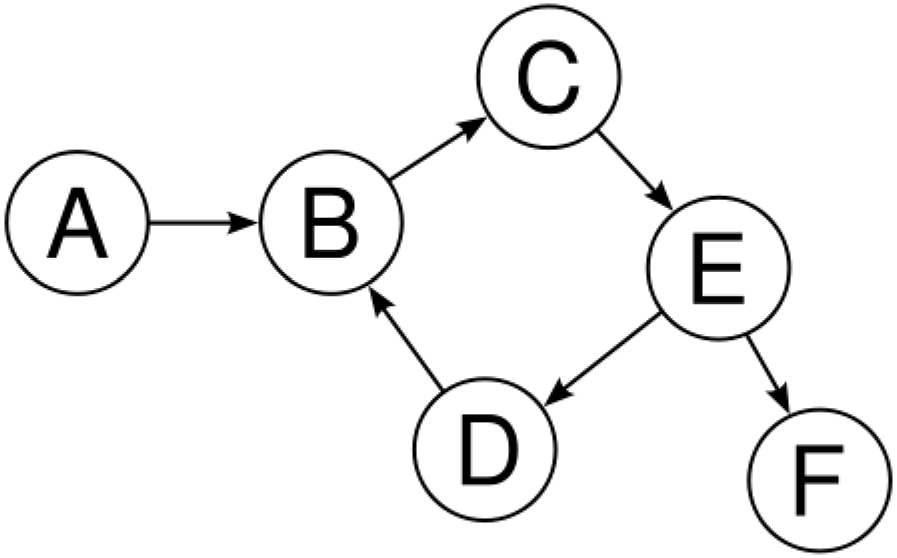
Graphs

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), (**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_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 $\leq$ 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 $R^{i-1} \subseteq R^i, i = 2, 3, \ldots, n - 1$. 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 $a \equiv a$ (mod m)}? *yes*
- *Symmetric* – is it true that if $a \equiv b$ (mod m) then $b \equiv a$ (mod m)? *yes*
- *Transitive* - is it true that if $a \equiv b$ (mod m) and $b \equiv c$ (mod m) then $a \equiv c$ (mod 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 <u>reflexive</u>.

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 <u>symmetric</u>.

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 <u>transitive</u>.

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”}

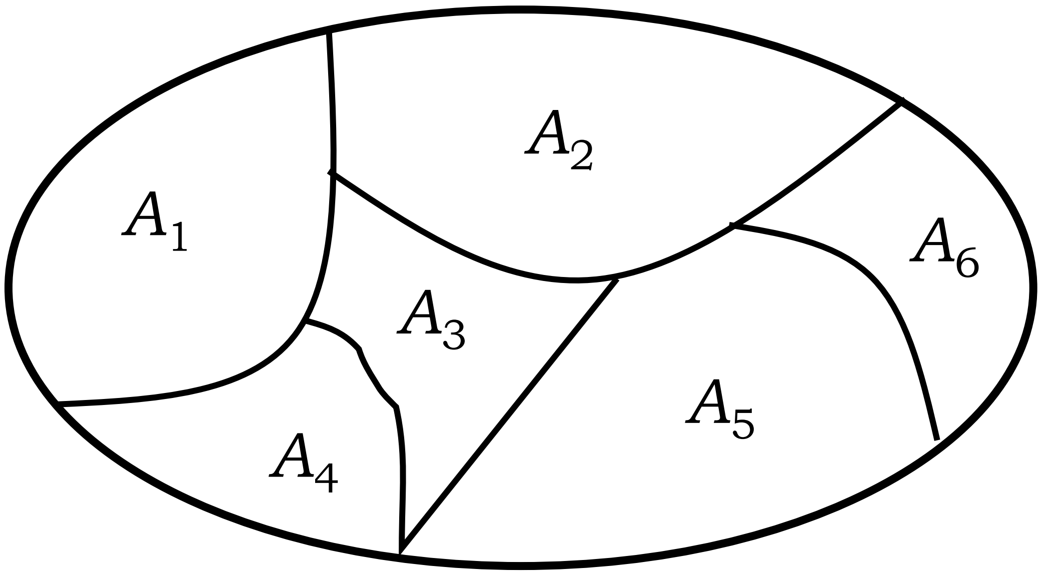
[“joe”] = {“joe”, “doe”, “bye”}

[“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 <u>**is**</u> a partial order, and (*A*, *R*) <u>**is**</u> 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?

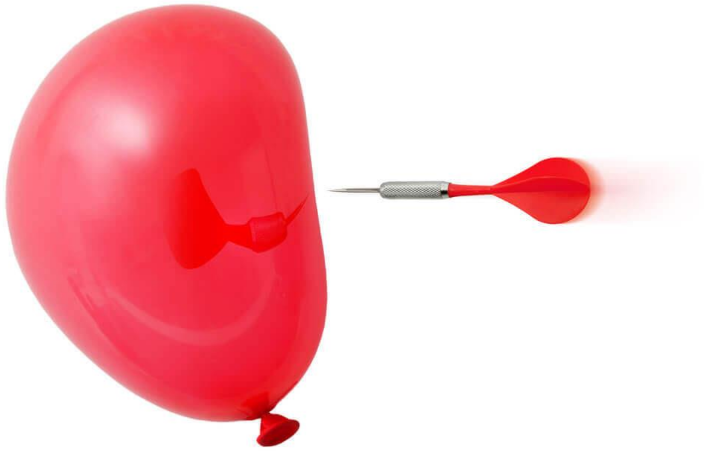
Every two elements of **Z** are comparable; that is, $a \leq b$ or $b \leq a$ for all integers.

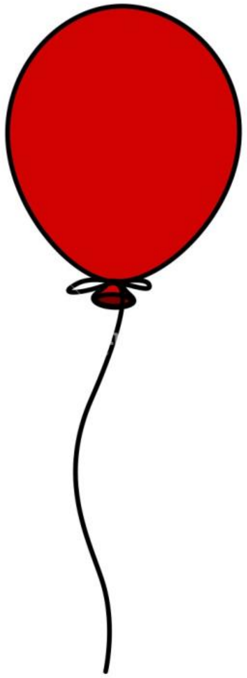
The poset $(Z^+, |)$ is not totally ordered where $|$ means divides. Why?

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 ≡ 188 (mod 10). 208 – 108 = 10×10 is divisible by 10

40 ≡ 14 (mod 13). 40 – 14 = 26 = 2×13 is divisible by 13

206 ≡ 342 (mod 17). 206 – 342 = -136 = -8×17 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: $Z_n - \{0\} \rightarrow Z_n - \{0\}$ 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

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

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

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

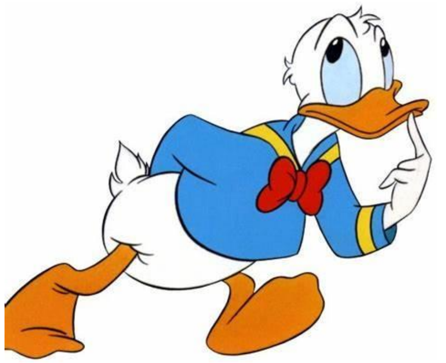
Example n = 7, b = 2

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

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

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

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 $0 \le d_i \le b-1, i = 0, \ldots, k, d_{k-1} \ne 0$, 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

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

$c, c_0, \ldots, c_{m-1}$, where $0 < c < b$ and $-b < c_i < b$ for $i = 0, 1, \ldots, m-1$. 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 $\gcd(a, b)$ 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.

$$\gcd(3000,7700) = 2^2 \times 5^2 = 100$$

- 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 $b \neq 0$ 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)

$\operatorname{lcm}(a, b)$ is the smallest multiple of both a and b.

Proposition. $\operatorname{lcm}(a, b) = \frac{ab}{\gcd(a,b)}$.

Example. $a = 585, b = 1035$

$$a \times b = 585 \times 1035 = 605475$$

$$\gcd(585,1035) = 45$$

$$\operatorname{lcm}(585,1035) = 23 \times 585 = 13 \times 1035$$

$$= 13455$$

$$= \frac{605475}{45}$$

Fraction in Simplest Form

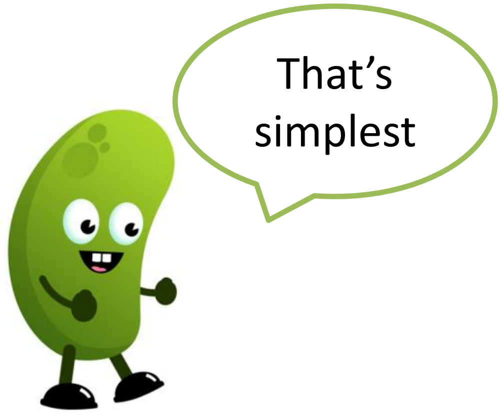
$$\frac{a}{b} \text{ in simplest form is } \frac{a/\gcd(a,b)}{b/\gcd(a,b)}$$

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 $g = \gcd(a, b) = sa + tb$ for $a = 6700, b = 3000$ obtaining $g = 100, s = 13, t = -29$.

PSN. Solve $g = \mathrm{gcd}(a, b) = sa + tb$ 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

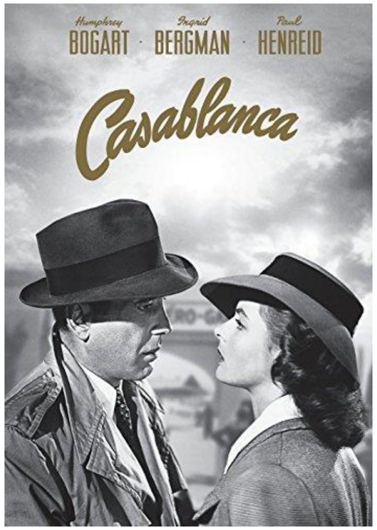
```
function ExtEuclidGCD(a,b,g,s,t)
    Input: a, b (nonnegative integers)
    Output: return g = gcd(a,b) and integers s and t such that sa + tb = g
      if (b == 0)    //BOOTSTRAP CONDITION
           g = a
           s = 1
           t = 0
      else
           r = a mod b
           q = a/b
           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."



Rivest-Shamir-Adelman (RSA) Public-Key Cryptosystem

RSA is a widely used public key cryptosystem.

Public Key Cryptosystem

- Pioneered by Diffie and Hellman.
- Compute a **public key** *E* and **private** key *D*, where *E* is used to encrypt messages and *D* is be used to decrypt messages that have been encrypted using *E*.
- These keys need to be chosen so that it is computationally infeasible to derive *D* from *E*. Anyone with the public key *E* is able to encrypt a message, but only someone knowing *D* is able (in real time) to decrypt an encrypted message.
- Many public-key cryptosystems have been designed. In this course we cover the popular **RSA** public key cryptosystem due to Rivest, Shamir, and Adleman.

RSA Cryptosystem

1. Compute two large primes p and q and set $n = pq$.

2. The Euler Totient Function $\varphi(n)$ is the number of positive integers less than n that are co-prime (relatively prime) to n

   Chose the public key e to be a positive integer that is relatively prime to $\varphi(n) = (p - 1)(q - 1)$, i.e., $\gcd(e,\varphi(n)) = 1$.

3. Computer the private key using formula

$$d = e^{-1} \pmod{\varphi(n)}$$

In PSN we used the Principle of Inclusion-Exclusion to show that

$$\varphi(n) = (p - 1)(q - 1).$$

Solution to PSN Repeated Here

Let $U = \{1, \ldots, n\}, A = \{p, 2p, \ldots, qp\}, B = \{q, 2q, \ldots, pq\}$

The number of numbers in *U* that are divisible by either *p* or *q* is given by

$$|A \cup B| = |A| + |B| - |A \cap B| = q + p - 1$$

Since both *p* and *q* are prime, the number of numbers in *U* that are relatively prime to *n* equals the number of numbers in *U* that are neither divisible by *p* nor *q*, which is given by

$$\begin{aligned} |\overline{A \cup B}| &= n - |A \cup B| \\ &= n - (q + p - 1) \\ &= pq - q - p + 1 = (p - 1)(q - 1) \end{aligned}$$

Implementation of RSA – Computing Large Primes *p* and *q*

- Small primes can be computed quickly. But how to compute a large prime p involving for example 100 digits.
- The solution is to randomly generate the 100 digits and to test whether it is prime. There are efficient algorithm for doing this.
- The issue then becomes will a positive result occur, i.e., a prime be found, in reasonable time or are the prime numbers too sparse.
- It follows from one off the deepest theorems in all of mathematics known as the prime number theorem that they are not.

Prime Number Theorem

Let $\pi(n)$ be the number of primes less than or equal to n. The prime number theorem states that

$$\pi(n) \sim \frac{n}{\ln n}$$

i.e., $\pi(n)$ approaches $\frac{n}{\ln n}$ as $n \to \infty$.

For example, consider $n = 999{,}999$. $\ln(999{,}999) \approx 14$. This means more than 1/14 of the numbers less than 999,999 are prime. Therefore, if we test, say 50 randomly chosen 6-digit numbers, we have a good chance of hitting one that is prime.

For 100-digit number, $\ln n < 4 \times \log_{10} n \approx 400$. Therefore, if we test, say 1000 randomly chosen 100-digit numbers, we have a good chance of hitting one that is prime.

Implementation of RSA – Computing Private Key from Public Key

PSN. Describe algorithm for computing private key $d = e^{-1} \pmod{\Phi(n)}$

Hint

Show that private key $d = s$, where $se + t\varphi(n) = 1$. Then explain how to compute s.

Prime Factorization is a Hard Problem

- Why can't an eavesdropper also compute d by factoring n into the two primes p and q and then computing $\varphi(n) = (p-1)(q-1)$. Once this done the eavesdropper can use the Extended GCD algorithm to compute the private key d.
- Fortunately, the eavesdropper can't do so in real time because Prime Factorization is hard even for a number n that is the product of two primes.
- No algorithm is currently known that can do it in real time for integers that are hundreds of digits long.

RSA
Encryption and Decryption of Messages

Message *m* is encrypted using formula:

$$c \equiv m^e \ (\text{mod } n).$$

Encrypted message *c* is decrypted using formula:

$$m \equiv c^d \ (mod\ n).$$

Note to be able to recover m, m < n.

Theorem on which correctness of RSA is based

Theorem (Rivest-Shamir-Aldelman). Let $n = pq$ where p and q are two prime numbers, let e be an integer that is relatively prime with $\varphi(n)$, and let d be its multiplicative inverse mod $\varphi(n)$, that is, $ed \equiv 1 \pmod{\varphi(n)}$. Then, for any integer m,

$$m^{ed} \equiv m \pmod{n}.$$

Euler’s Totient Theorem

To prove the Theorem will need to apply a generalization of Fermat’s Little Theorem due to Euler called Euler’s Totient Theorem.

Theorem (Euler). Let *n* and *b* be relatively prime numbers. Then

$$b^{\varphi(n)} \equiv 1 \pmod{n} .$$

Note that $\varphi(n) = n - 1$ for n prime so we obtain Fermat’s Little Theorem as a corollary.

Proof of Euler's Totient Function

Let $Z_n^* = \{r_1, r_2, \ldots, r_{\varphi(n)}\}$ be the set of number between 1 and *n* – 1, inclusive, that are relatively prime to *n*. For example

$$Z_{12}^* = \{1,5,7,11\}$$

Let $b \in Z_n^*$. Then, b is invertible mod *n.*

b^{-1} can be computed using extended Euclid GCD.

Proof of Euler's Totient Theorem Similar to Proof of Fermat's Little Theorem

Since b is invertible mod n, it follows that $f(x) = bx \pmod{n}, x \in Z_n^*$ is a bijective function from Z_n^* to itself, so that

$$(br_1)(br_2)\cdots(br_{\varphi(n)}) \equiv r_1 r_2 \cdots r_{\varphi(n)} \pmod{n}$$

$$\Rightarrow b^{\varphi(n)} r_1 r_2 \cdots r_{\varphi(n)} \equiv r_1 r_2 \cdots r_{\varphi(n)} \pmod{n}$$

$$\Rightarrow b^{\varphi(n)} \equiv 1 \pmod{n}$$

Examples *n* = 12, *b* = 5 & *b* = 11

$$Z_{12}^{*} = \{1,5,7,11\}$$

$(5\times1) \times (5\times5) \times (5\times7) \times (5\times11) \equiv 5 \times 1 \times 11 \times 7 \pmod{12}$

$\Rightarrow 5^4 \times (1 \times 5 \times 7 \times 11) \equiv 1 \times 5 \times 7 \times 11 \qquad \pmod{12}$

$\Rightarrow 5^4 \equiv 1 \pmod{12}$

$(11\times1) \times (11\times5) \times (11\times7) \times (11\times11) \equiv 11 \times 7 \times 5 \times 1 \pmod{12}$

$\Rightarrow 11^4 \times (1 \times 5 \times 7 \times 11) \equiv 1 \times 5 \times 7 \times 11 \qquad \pmod{12}$

$\Rightarrow 11^4 \equiv 1 \pmod{12}$

Proof of Theorem

Since $ed \equiv 1 \pmod{\varphi(n)}$, it follows that

$$ed = \varphi(n)k + 1,$$

for some integer k.

Proof for case when *m* and *n* are relatively prime

Applying $ed = \varphi(n)k + 1$ from previous slide and Euler's Totient Theorem

$$\begin{aligned} m^{ed} &= m^{\varphi(n)k + 1} \\ &= (m^{\varphi(n)})^k m \\ &\equiv (1)^k m \pmod{n} \\ &= m \pmod{n} \end{aligned}$$

Prime Number Dilemma

Should you say "All prime numbers are odd except one“?

Or "All prime numbers are odd except two?"

Graph Theory

Reading:

Textbook: Chapter 6, pp. 331-340

Supplemental Notes on Graphs and Networks

Definition of a graph

- A graph G is a mathematical structure defined by a set of **vertices** (also called **nodes**) $V = V(G)$ and a set of edges $E = E(G)$, where each edge e is an unordered pair $\{a,b\}$ of **distinct** vertices.
- Edge e can also be denoted by ab. We say that e is **incident** with its **end vertices** a and b and say that a and b are **adjacent**.
- We denote the number of vertices and edges by $n = n(G)$ and $m = m(G)$, respectively, i.e., $n = |V|$ and $m = |E|$.
- The number n of vertices is the **order** of G and the number m of edges is the **size** of G.

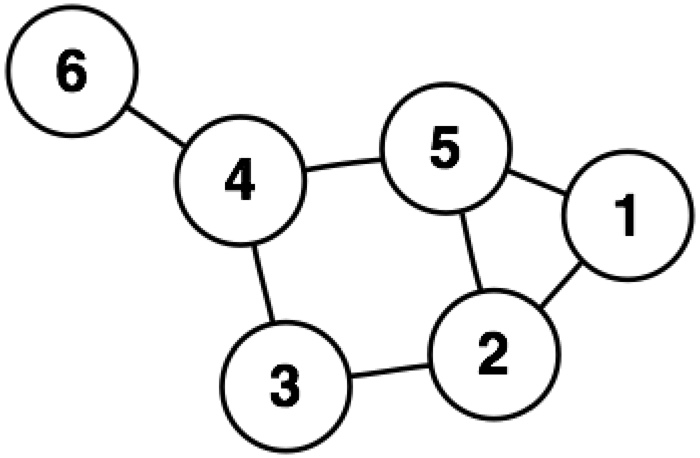
Drawing in Plane

Graphs can be drawn in the plane where vertices are represented as points and two points are joined with a line or curve if their corresponding vertices are adjacent.

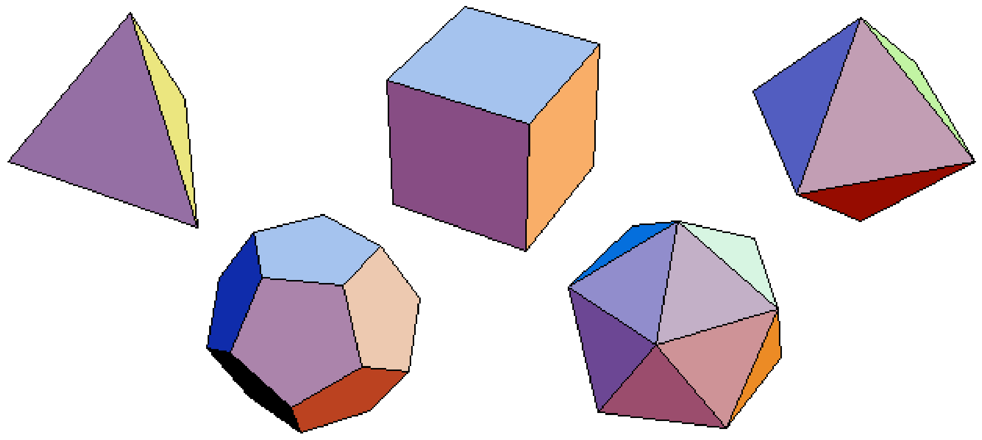
For example the graph *G* = (*V*,*E*) where

V = {1,2,3,4,5,6} and E = {12, 15, 23, 25, 34, 45, 46}

can be drawn is the plane as follows.

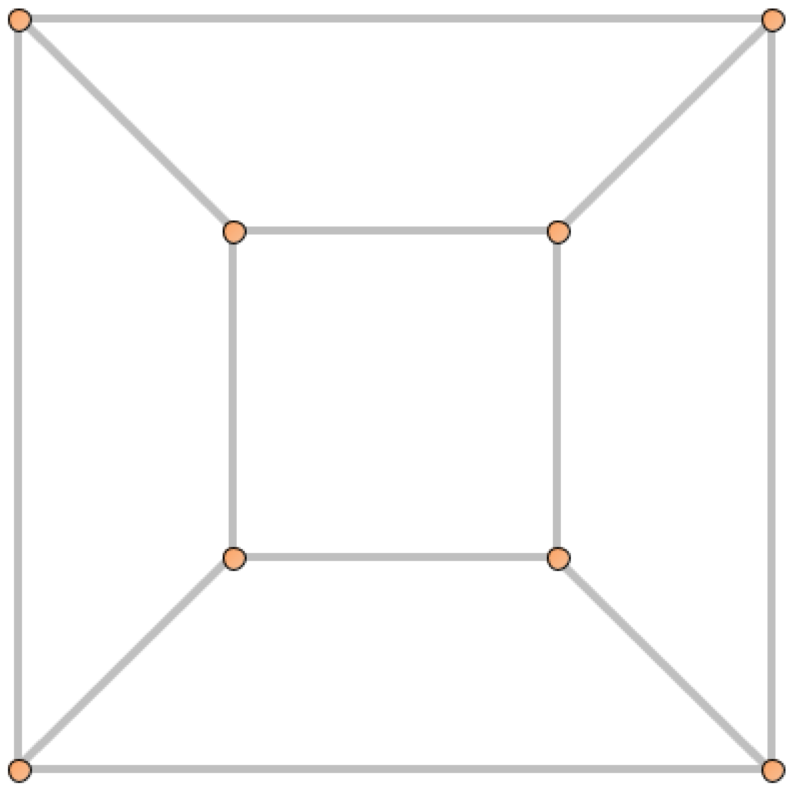


What is origin of the term vertex on edge



Above are the regular polyhedra that were studied by Euler. They are represented by graphs where the vertices of the polyhedron correspond to the vertices of the graph and the edges of the polyhedron correspond to the edges of the graph.

Graph representing cube



Modeling Networks

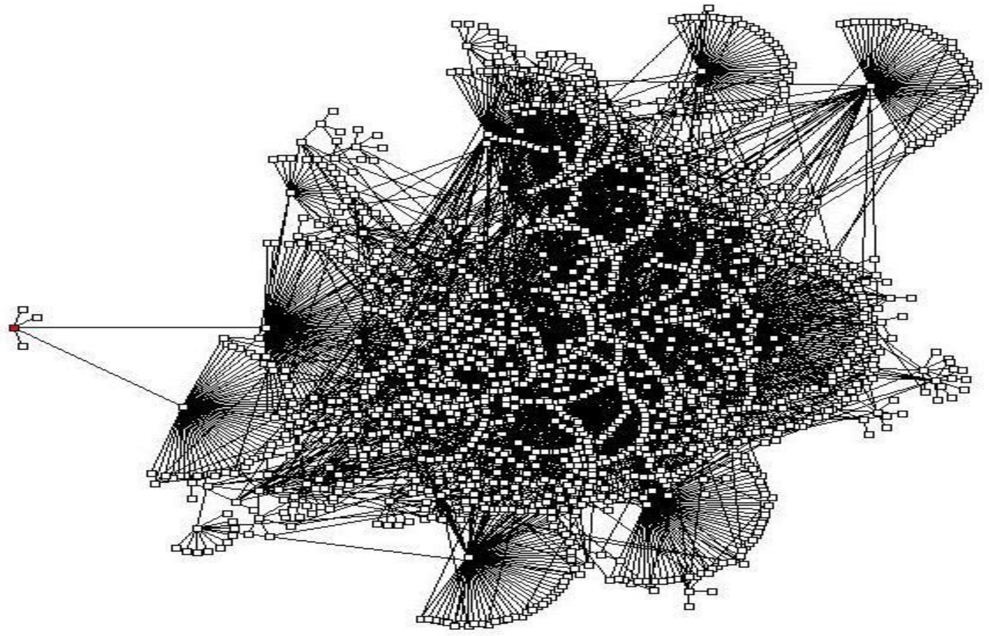
Graphs play a fundamental role in the design and analysis of networks and their associated algorithms and protocols, since they model the underlying topologies on which networks are built.

Discovery of fundamental properties and characteristics by mapping out topology of P2P network

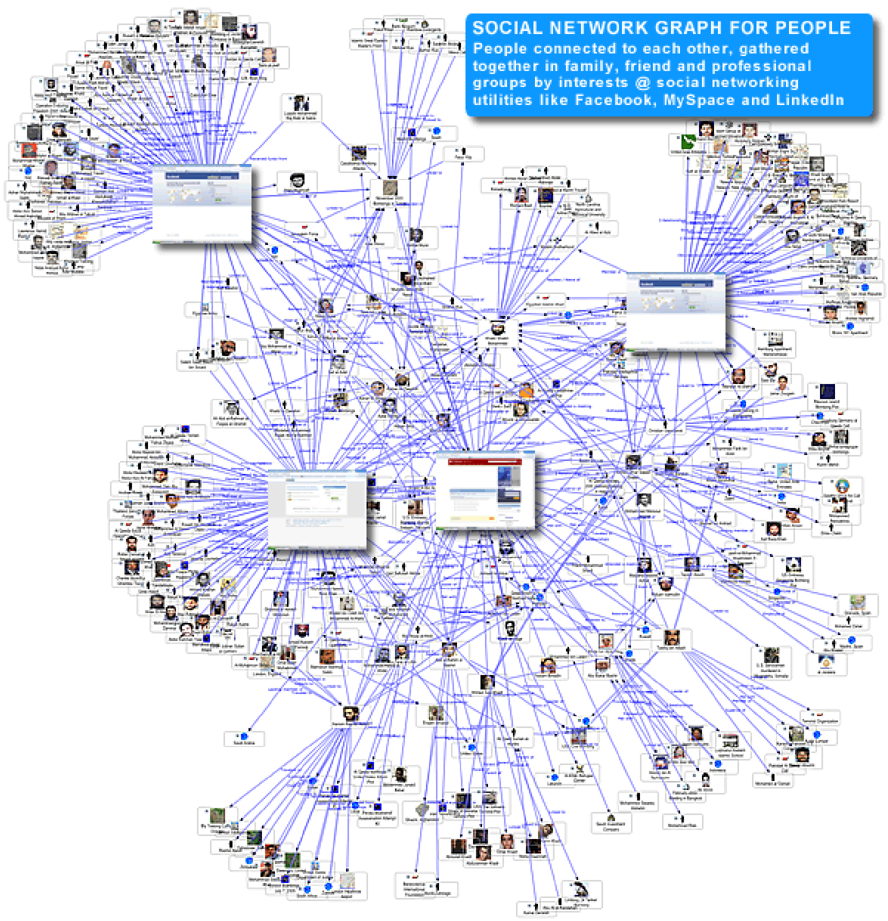
Designed a Parallel Network Crawler:

Discovered small world property, power laws and "short-circuiting" phenomenon

A snapshot of portion of Gnutella P2P network



Graph of Social Network


SOCIAL NETWORK GRAPH FOR PEOPLE
People connected to each other, gathered together in family, friend and professional groups by interests @ social networking utilities like Facebook, MySpace and LinkedIn

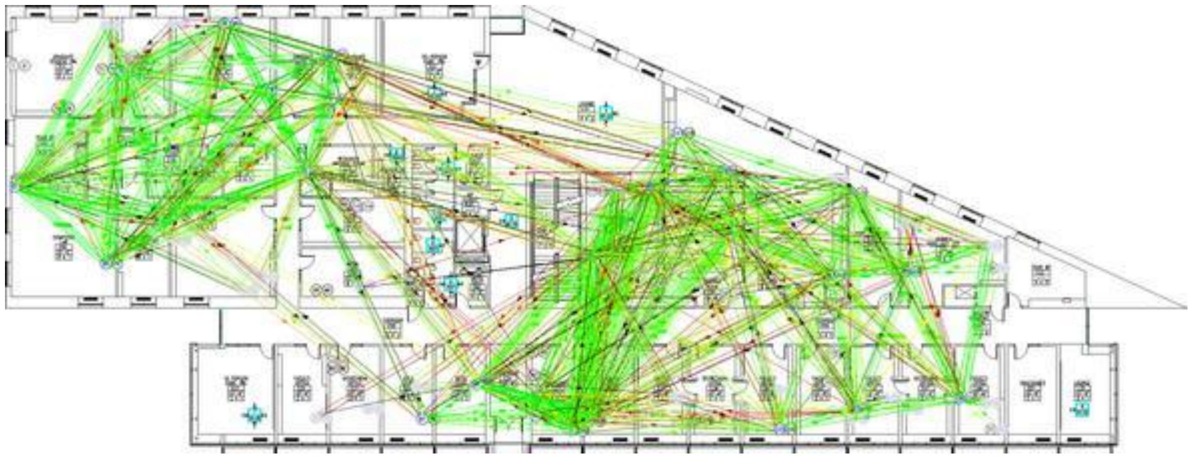
Sensor Networks

Simulations using Qualnet, ns-2

CDMC Center at UC: Crossbow TelosB Motes testbed

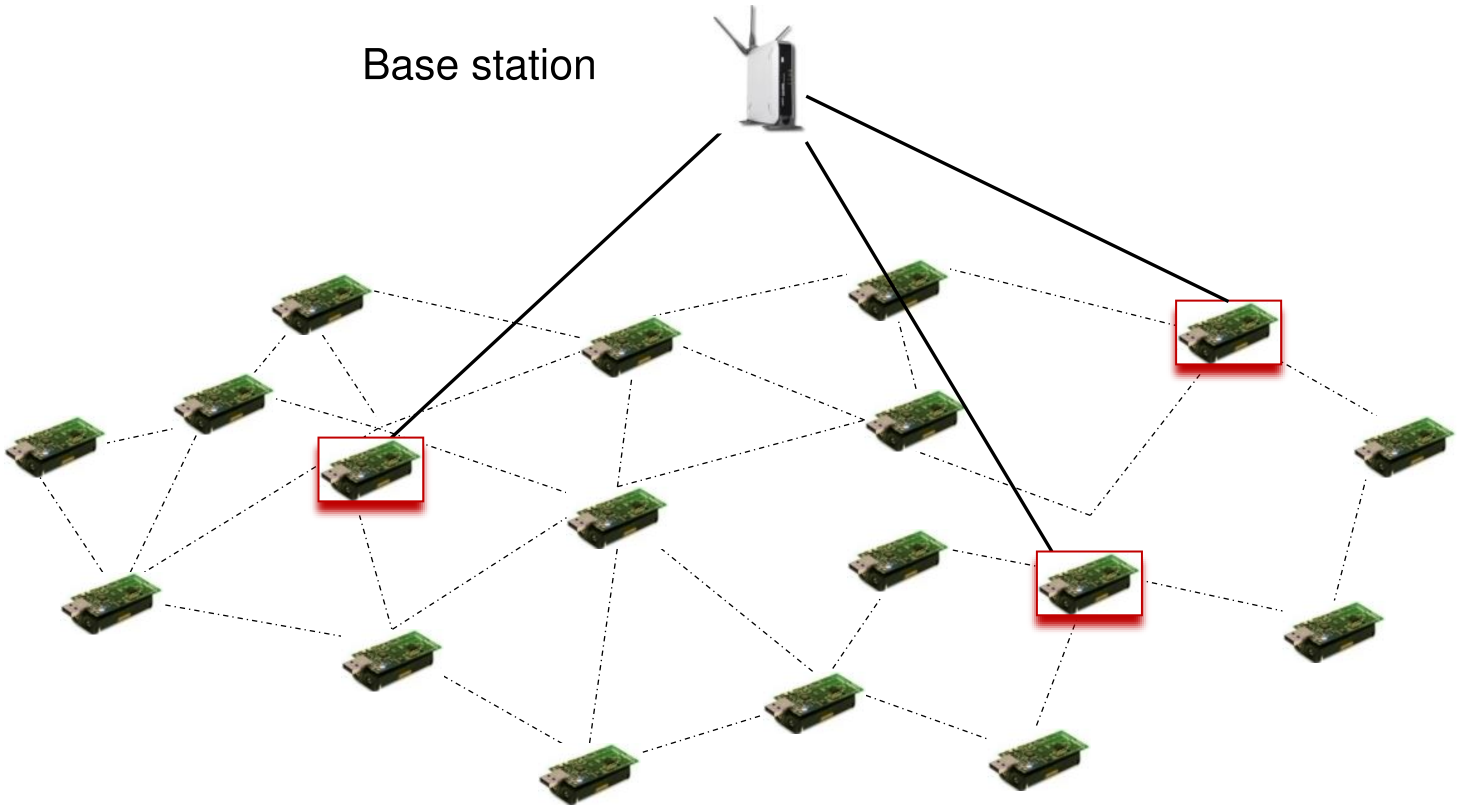
MoteLab testbed at Harvard

<u>TelosB Mote</u>



<u>One of the three floor maps of MoteLab</u>

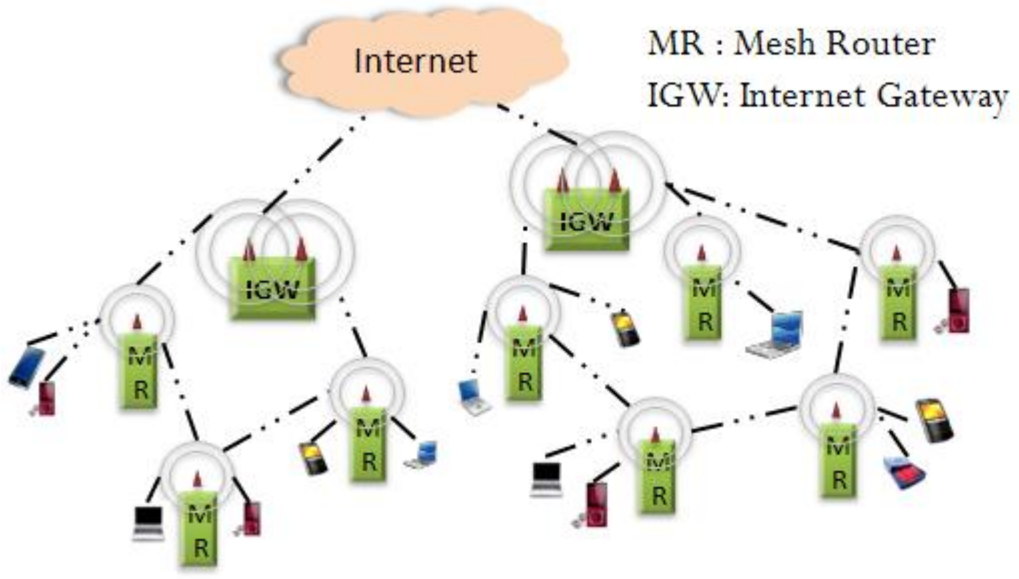
Wireless Sensor Networks



Wireless Mesh Network

CDMC Wireless Mesh Network

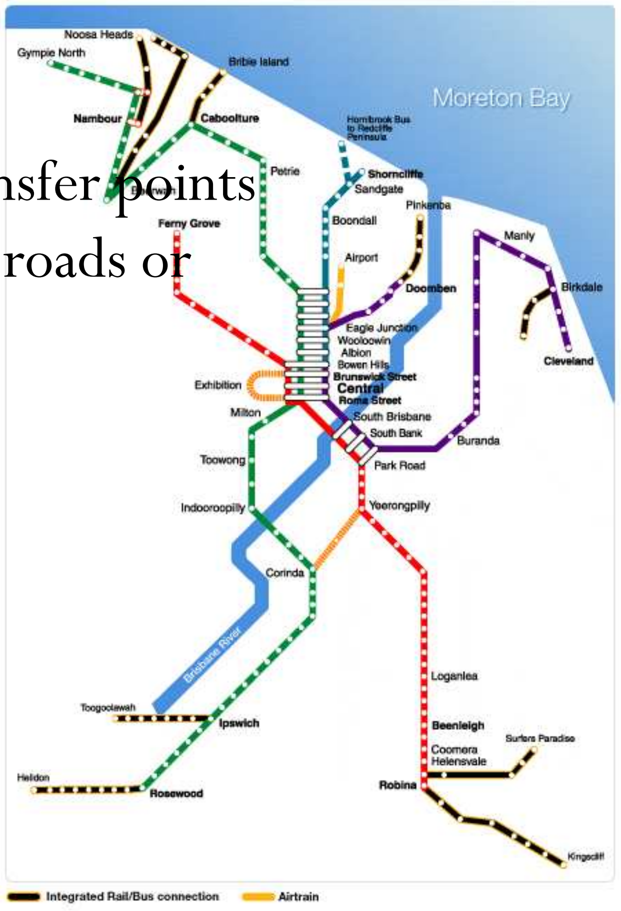
testbed at UC



Architecture of a Wireless Mesh Network

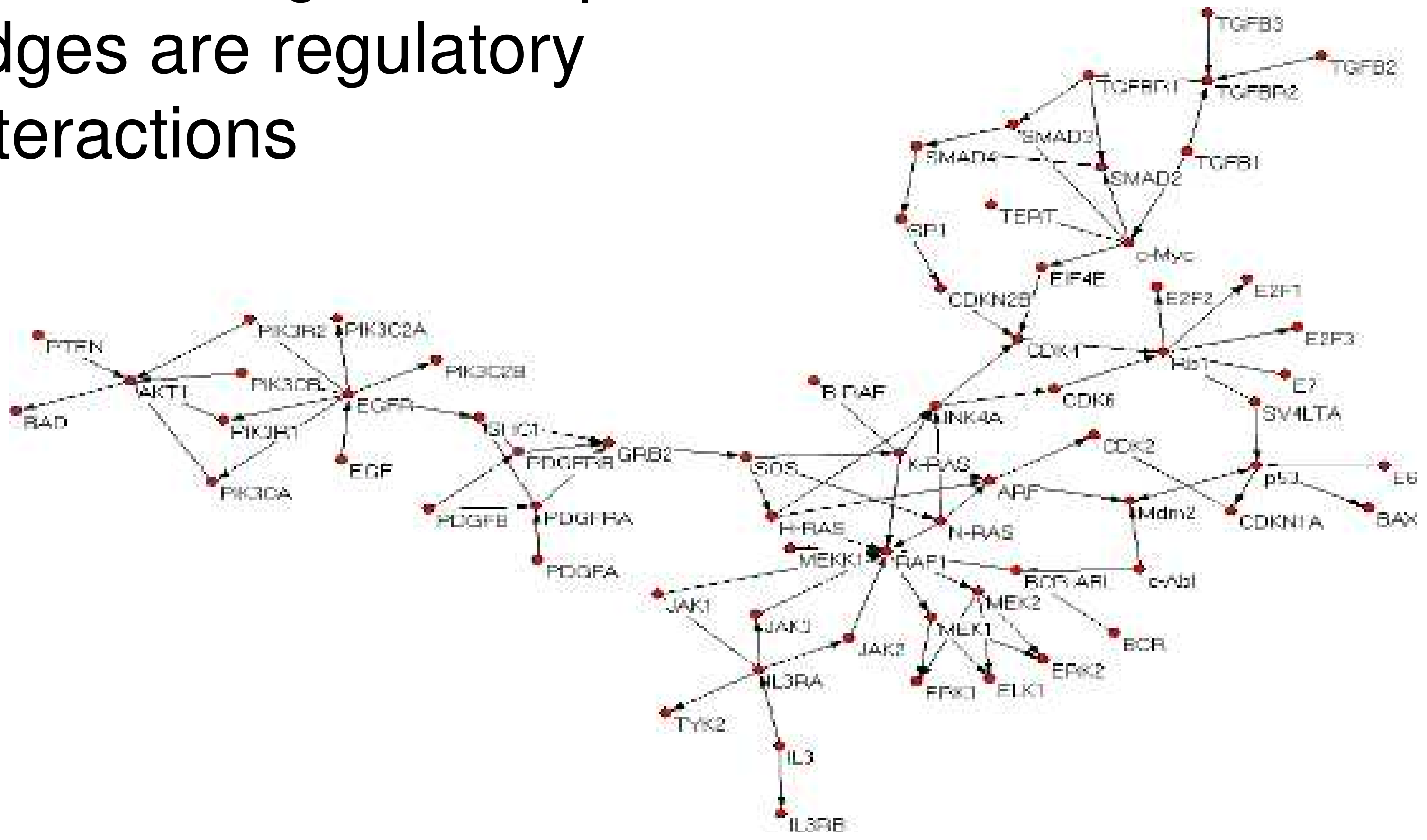
Transportation Network

– Nodes are cities, transfer points or depots, edges are roads or transport routes



Genetic regulatory networks

Nodes are genes or proteins, edges are regulatory interactions



Network Service

Given:

Set of disconnected peers

Set of public routers (e.g., PlanetLab)

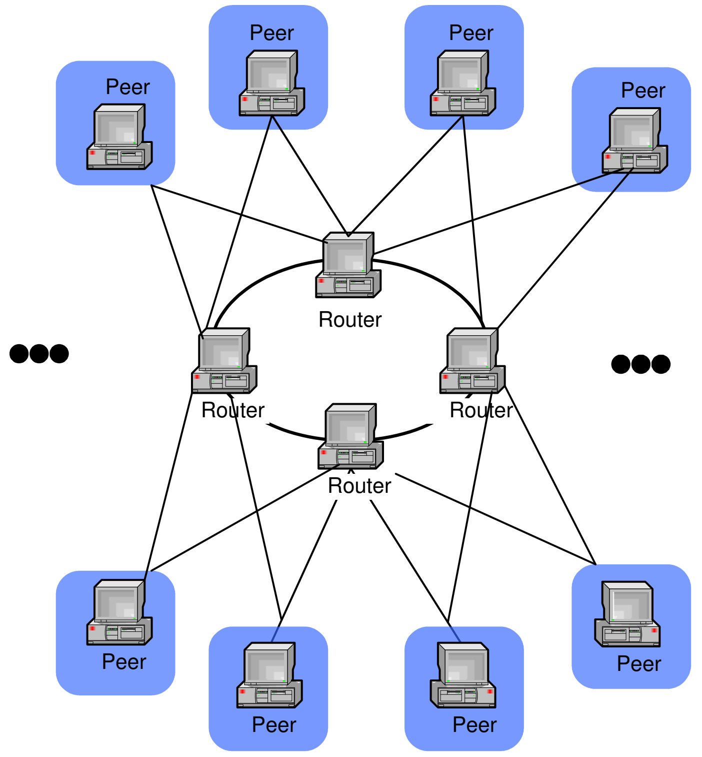
Each peer connects to Δ closest routers

Workload:

Peer-to-Peer file transfer

Problem:

How to maximize throughput in an online way (no global knowledge of demands) ?



There are many more examples of graphs modeling networks.

Extracurricular Exercise. Search the web for more examples.

Vertex Degrees

The *degree* of a vertex $v \in V$, denoted by $deg(v)$, is the number of edges incident with v.

Let $\delta = \delta(G)$ denote the minimum degree over all the vertices

Let $\Delta = \Delta(G)$ denote maximum degree over all the vertices

Then,

$$\delta(G) \leq deg(v) \leq \Delta(G)$$

Euler's Degree Formula

Theorem. The sum of the degrees over all the vertices is twice the number of edges, i.e.,

$$\sum_{v \in V} deg(v) = 2m\,.$$

Proof of Euler’s Degree Formula

Every edge is incident with exactly two vertices. Therefore, when summing the degrees over all the vertices we count each edge exactly twice, once for each of its end vertices.

Corollary. The number of vertices of odd degree is even.

This follows from Euler’s Degree Formula and the fact that a sum of numbers is even number if and only if an even number of these numbers is odd.

PSN. Obtain a formula for the average degree α of a vertex in terms of the number of vertices n and the number of edges m.

Hint. Use Euler's Degree Formula.

The average degree is at least as great as the minimum degree and no greater than the maximum degree, i.e.,

$$\delta \leq \alpha \leq \Delta.$$

Thus, we have

$$\delta \leq \frac{2m}{n} \leq \Delta$$

Regular graphs

A graph G is r-regular if each vertex has the same degree r.

Proposition. If G is an r-regular graph with n vertices and m edges, then

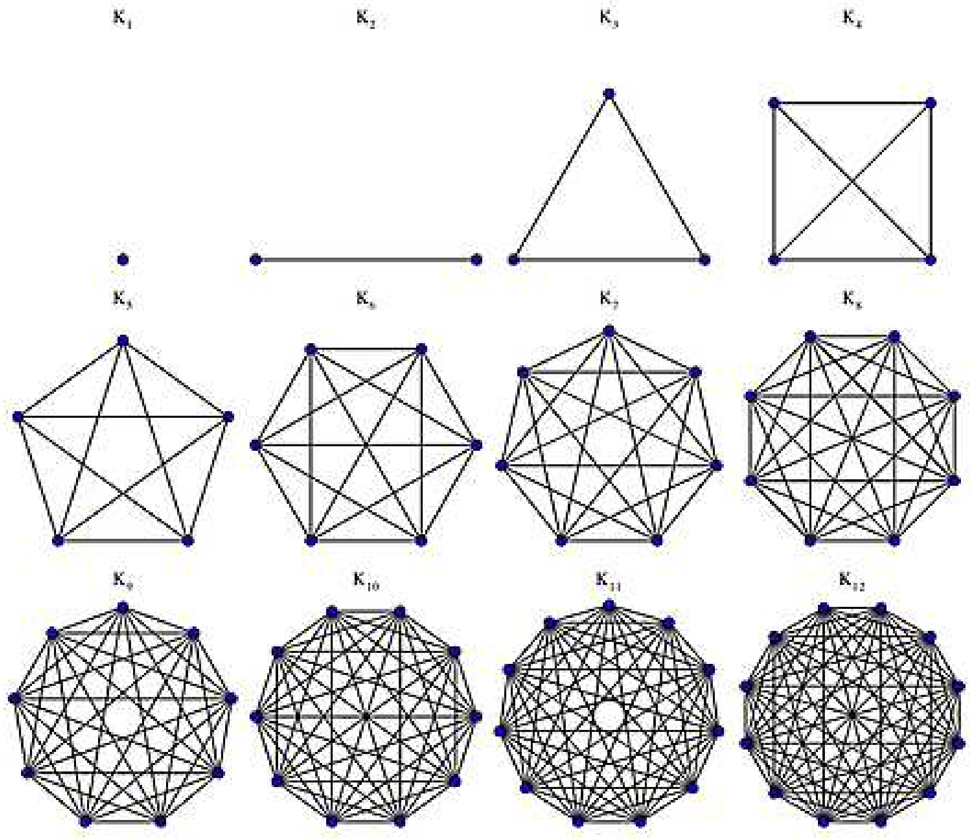
$$m = \frac{nr}{2}.$$

Proof. It follows from the definition of r-regular that $deg(v) = r, \forall v \in V$. Using Euler's degree formula we have

$$m = \frac{1}{2} \sum_{v \in V} deg(v) = \frac{1}{2} \sum_{v \in V} r = \frac{nr}{2}$$

Complete graph

The **complete graph** on n vertices denote by K_n is the graph where every pair of vertices are adjacent.

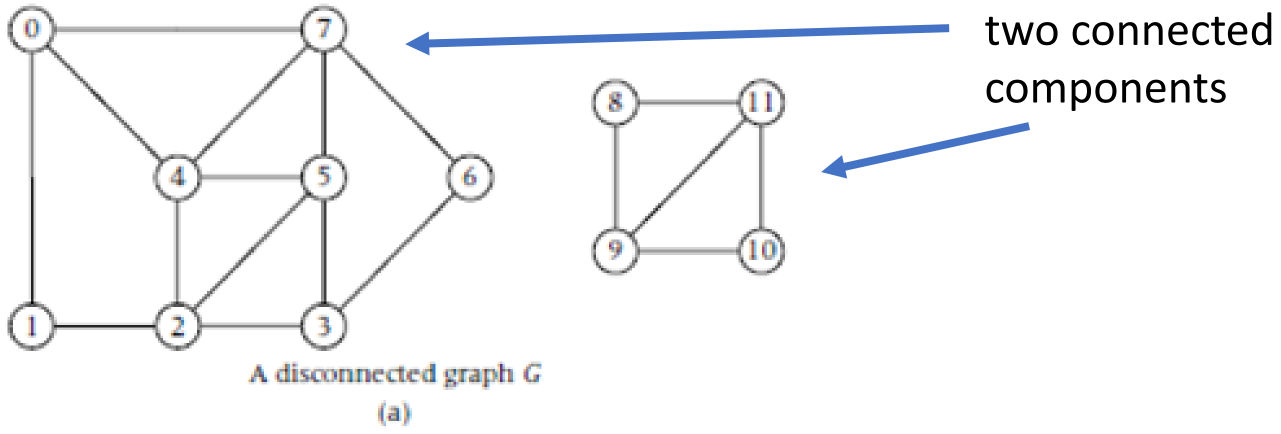


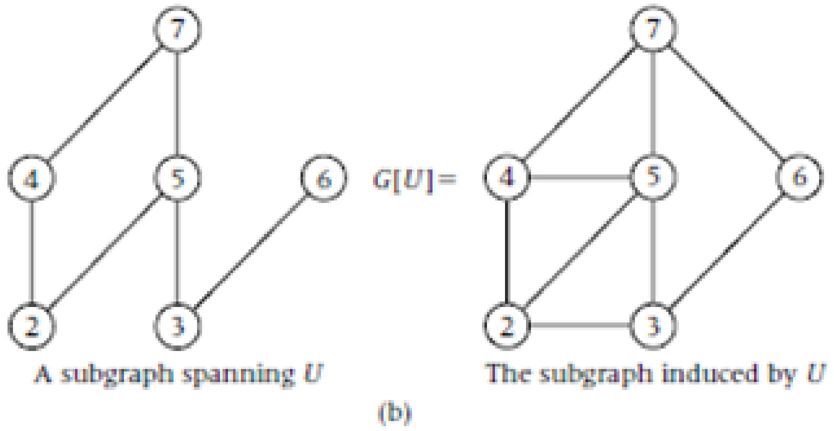
PSN. How many edges does the complete graph K_n have?

Subgraphs

- A *subgraph* *H* of *G* is a graph such that $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$.
- A subgraph *H* that is a tree is a *subtree* of *G*, or simply a *tree* of *G*.
- A subgraph *H* of *G* is called a *spanning* subgraph if *H* contains all the vertices of *G*. When a subgraph *H* of *G* is a tree, we use the term *spanning tree* of *G*.
- Given a subset *U* of vertices of *G*, the subgraph *G*[*U*] *induced by U* is the subgraph with vertex set *U* and edge set consisting of all edges in *G* having both end vertices in *U*.
- A component of a graph *G* is a connected induced subgraph of *G* that is not contained in a strictly larger connected subgraph

Sample Graph *G*
U = {2, 3, 4, 5, 6, 7}

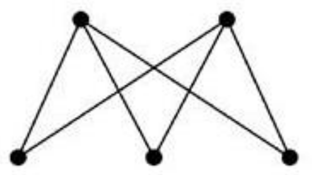


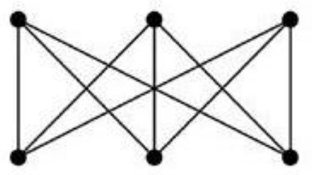


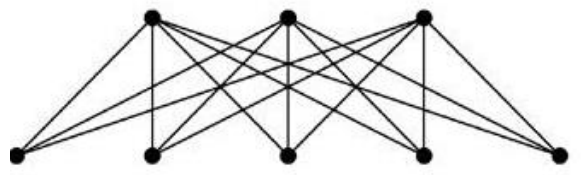
Bipartite graphs

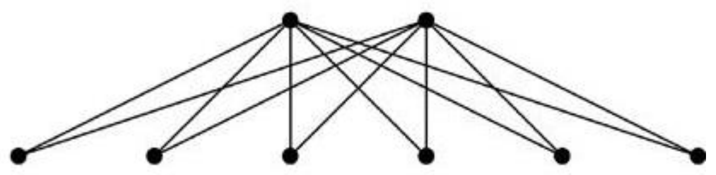
A graph is *bipartite* if there exists a bipartition of the vertex set V into two sets X and Y such that every edge has one end in X and the other in Y.

The *complete bipartite* graph $K_{i,j}$ is the bipartite graph with $n = i + j$ vertices, i vertices belonging to X and j vertices belonging to Y, such that there is an edge joining every vertex $x \in X$ to every vertex $y \in Y$.

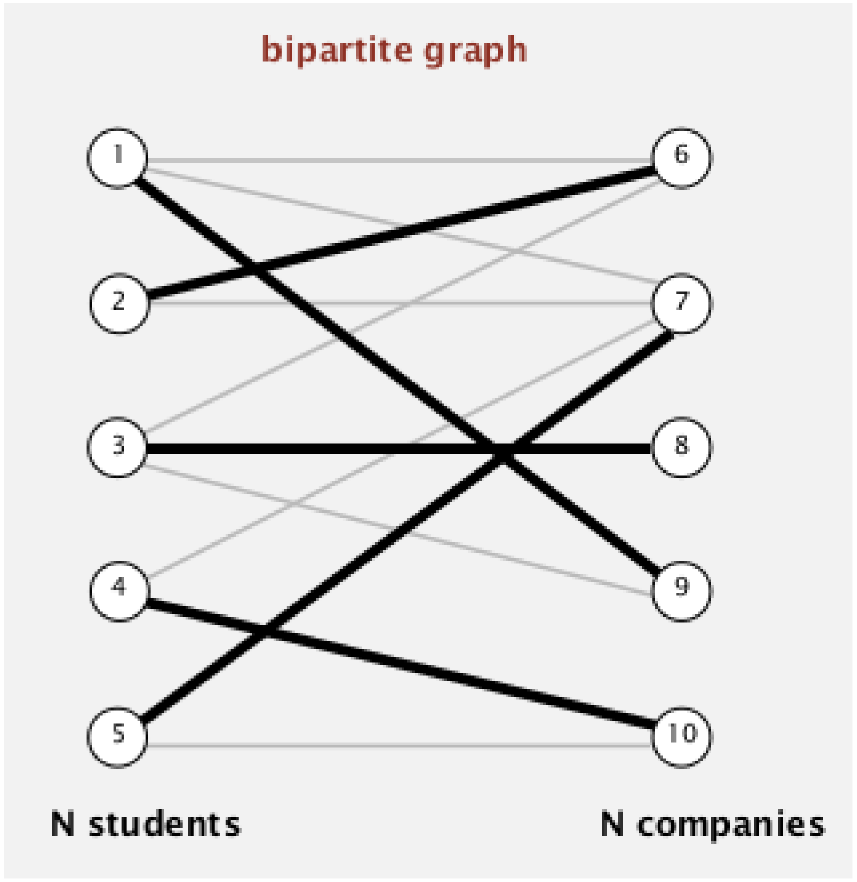

$K_{2,3}$


$K_{3,3}$

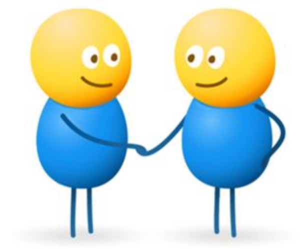

$K_{3,5}$


$K_{2,6}$

Matching Students to Companies



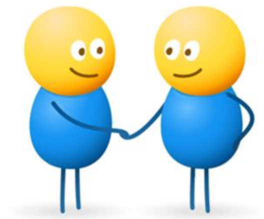
Handshaking Problem



UC students get together for a Bearcat reception and shake hands (pre-COVID). Is there a scenario where everyone has shaken a different number of hands from everyone else?

Surprisingly the answer is no. No matter who shakes hands in a group of people, there must be two people who have shook exactly the same number of hands!

Handshaking Theorem



Handshaking Theorem. Let G be a graph with at least two vertices. Then, at least two vertices have the same degree.

Proof.

- If there is two isolated vertices then we have two vertices both have degree 0 and we are done.
- If there is only one isolated vertex remove it. Now we have n vertices having n – 1 possible degrees.

Proof cont'd

- Take the n vertices to be pigeons and the possible degrees 1, 2, …, n – 1 to be pigeonholes.
- By the Pigeonhole Principle two pigeons are in the same pigeonhole, i.e., two vertices have the same degree.

Graph Isomorphism, Paths, Connectivity

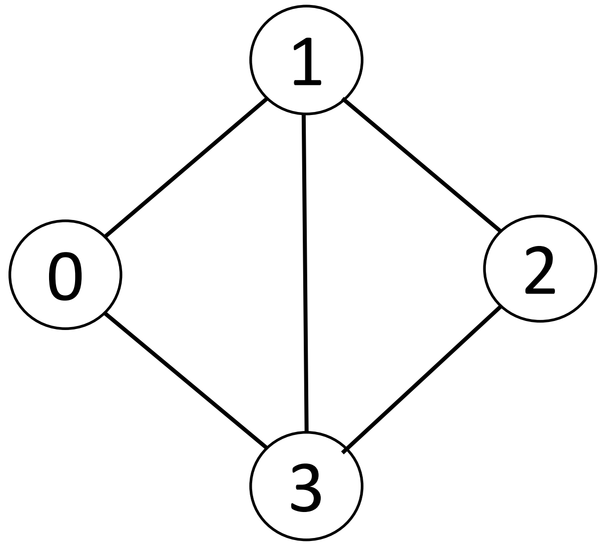
Reading:

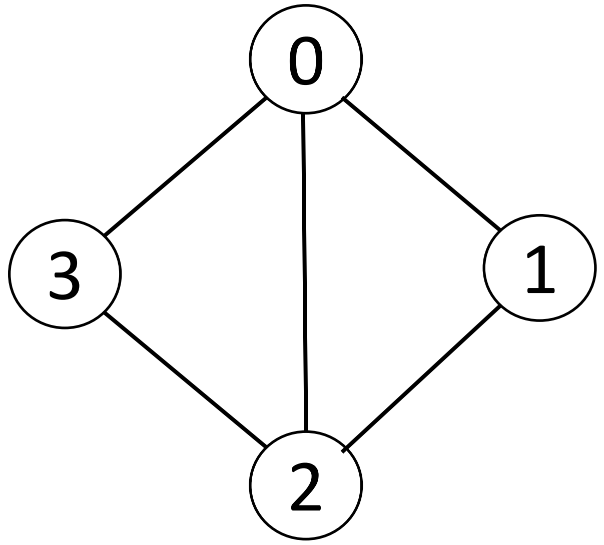
Textbook: Chapter 6, Sections 6.3, 6.4 pp. 340-346
Section 6.7 pp. 352-357

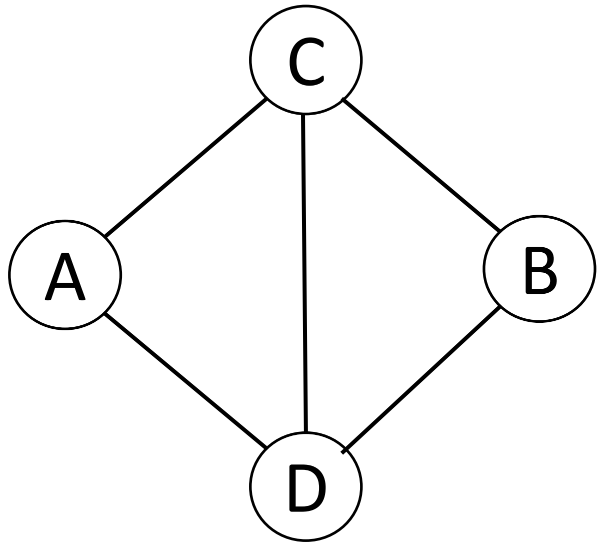
Supplemental Notes

Graph Isomorphism

Consider the three graphs below. Same if labels removed. They are not the same graph but they are **isomorphic** graphs.







$V = \{0,1,2,3\}$
$E = \{01,03,12,13,23\}$

$V = \{0,1,2,3\}$
$E = \{01,02,03,12,23\}$

$V = \{A,B,C,D\}$
$E = \{AC,AD,BC,BD,CD\}$

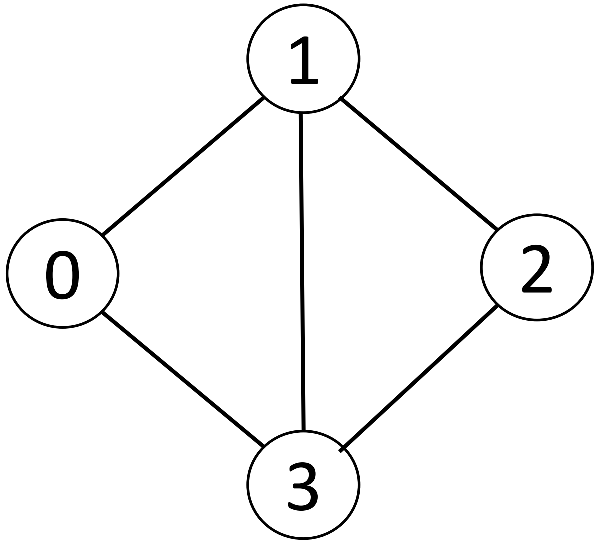
Graph Isomorphism

Two graphs G = (V,E) and G'= (V',E') are **isomorphic** if there exists a bijective mapping β: $V \to V'$ from the vertex set V of G onto the vertex set V' of G' such that adjacency relationships are preserved, that is,

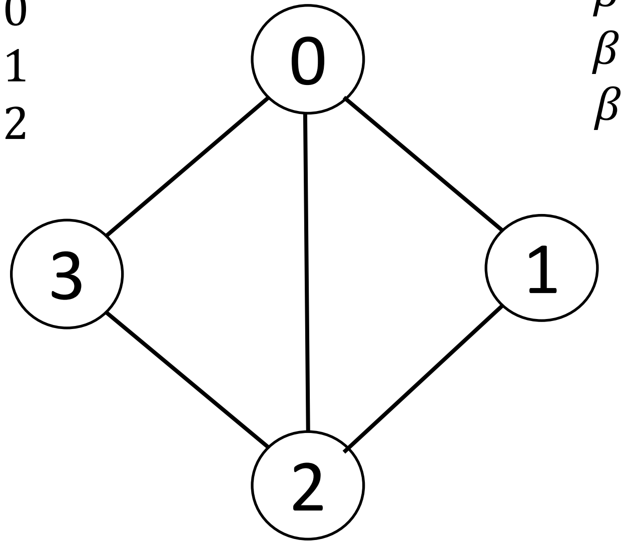
$$\{u,w\} \in E \text{ iff } \{\beta(u),\beta(w)\} \in E'.$$

The mapping β is called an **isomorphism**. Deciding whether two graphs are isomorphic is, in general, a difficult problem.

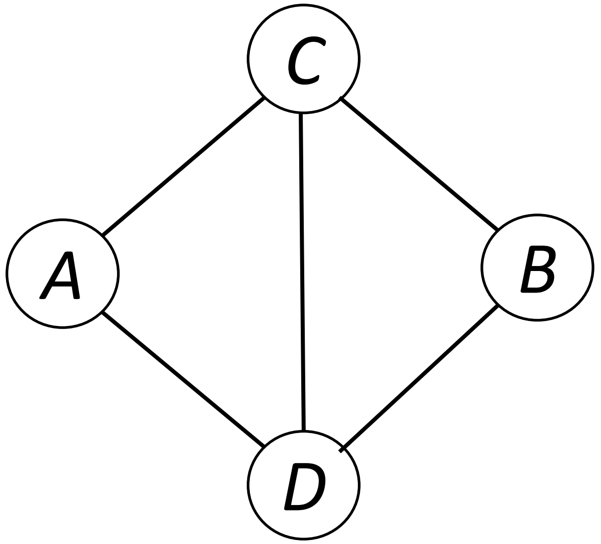
Isomorphisms for sample graphs



$\beta(0) = 3$
$\beta(1) = 0$
$\beta(2) = 1$
$\beta(3) = 2$



$\beta(0) = C$
$\beta(1) = B$
$\beta(2) = D$
$\beta(3) = A$



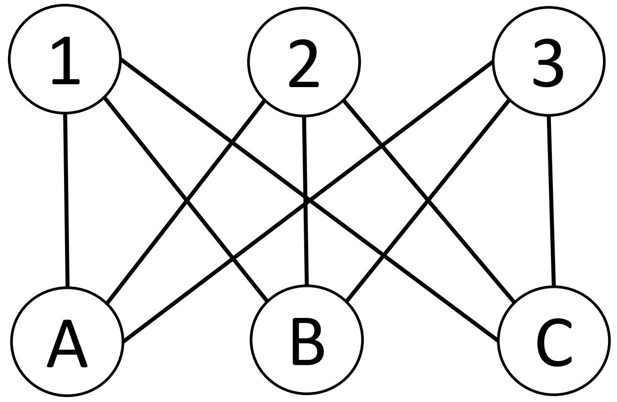
$V = \{0,1,2,3\}$
$E = \{01,03,12,13,23\}$

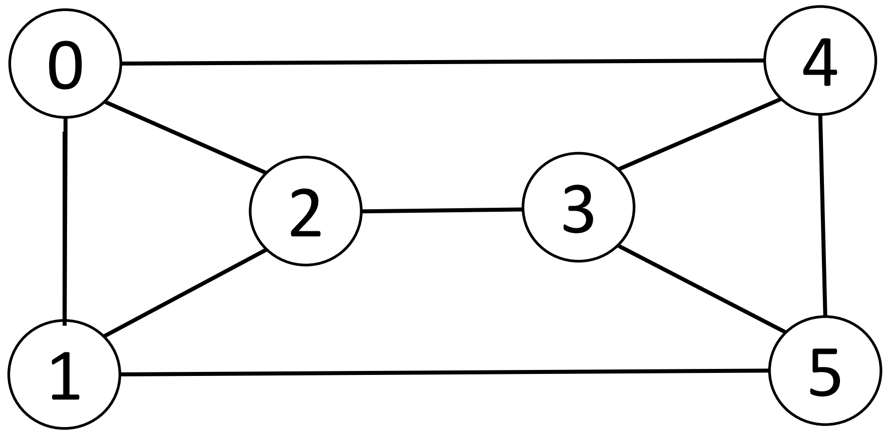
$V = \{0,1,2,3\}$
$E = \{01,02,03,12,23\}$

$V = \{A,B,C,D\}$
$E = \{AC,AD,BC,BD,CD\}$

Degrees and Isomorphism

If two graphs having different degree sequences, where the **degree sequence** is the non-increasing sequence of the vertex degrees, then they are not isomorphic. The converse is not true. It is possible for two graphs to have exactly the same degree sequence and not be isomorphic. For example, the graphs below each have 6 vertices of degree 3, but they are not isomorphic because the first graph $K_{3,3}$ is bipartite and the other graph contains a triangle so is not bipartite.





Graph Isomorphism Problem is Hard

No one has discovered a polynomial time algorithm for solving the graph isomorphism problem, although it is not NP-complete.

Paths

- A **path *P* of length *p*** ($p \geq 0$) joining vertices u and v is an alternating sequence of $p + 1$ vertices and p edges $u_0e_1u_1e_2 \ldots e_pu_p$ such that $u = u_0$, $v = u_p$, where e_i joins u_{i-1} and u_i, $i = 1,2, \ldots, p$.
- We call u_0 and u_p the **initial** and **terminal** vertices, respectively, and the remaining vertices in the path the **interior** (or **internal**) vertices.
- Vertices in a path can be repeated, but **edges must be distinct**.
- If $u_0e_1u_1e_2 \ldots e_pu_p$ includes repeated edges it is called a **walk**.
- In the textbook and literature when vertices are repeated, but not edges, it is sometimes called a **trail.** Other times it is called a **path** and a **simple path** refers to one where vertices are not repeated. This is not consistent in the literature.
- Since the path P is completely defined by the sequence of vertices $u_0u_1 \ldots u_p$, we often use this shorter sequence to denote P.
- If $u = v$, then the path is called a **closed path** or **circuit**.
- A **simple circuit** or **cycle** is a simple closed path.

Important special types of paths and cycles

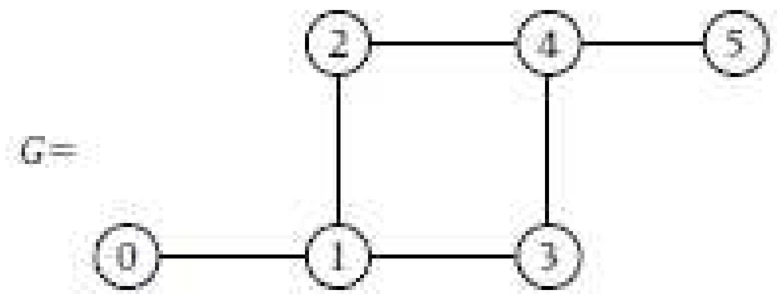
- A path that contains every edge in the graph exactly once is called an **Eulerian path (or Eulerian trail)**.
- A circuit that contains every edge exactly once is called an **Eulerian circuit** or **Eulerian tour**.
- A simple path that contains every vertex in the graph is called a **Hamiltonian path**.
- A cycle that contains every vertex in the graph is called a **Hamiltonian cycle**.

Distance and Diameter

- The **distance** between u and v, denoted by $d(u,v)$, is the length of a path from u to v that has the shortest (minimum) length among all such paths.
- By convention, if u and v are not connected, then $d(u,v) = \infty$.
- The **diameter** of G is the maximum distance between any two vertices.

Distance between every pair of vertices and the diameter for a sample graph *G*

We show the distances using a 6-by-6 **distance matrix**, whose *ij*th entry is given by $d(i,j)$.



Distance matrix of G

$$\begin{pmatrix} 0 & 1 & 2 & 2 & 3 & 4 \\ 1 & 0 & 1 & 1 & 2 & 3 \\ 2 & 1 & 0 & 2 & 1 & 2 \\ 2 & 1 & 2 & 0 & 1 & 2 \\ 3 & 2 & 1 & 1 & 0 & 1 \\ 4 & 3 & 2 & 2 & 1 & 0 \end{pmatrix} \quad \text{Diameter } G = 4$$

Connectedness

Two vertices $u, v \in V$ are *connected* if there exists a path (of length 0 when $u = v$) that joins them.

PSN. Show that the relation R where

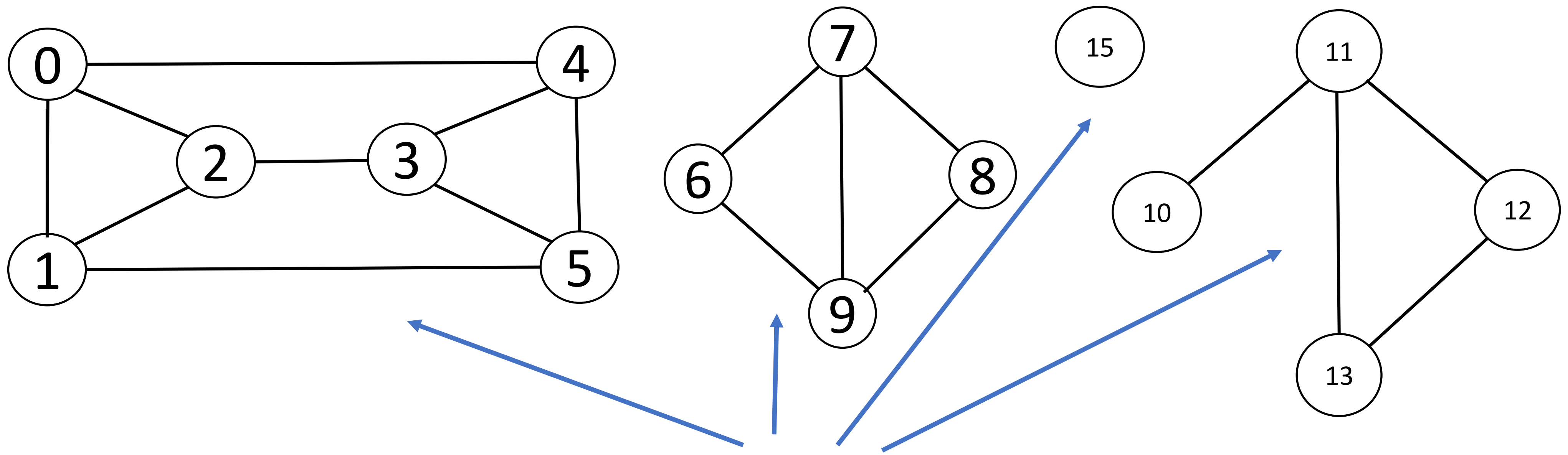
$$uRv \text{ iff } u \text{ is connected to } v$$

is an equivalence relation on V.

Connected Components

Each equivalence class *C* of vertices, together with all incident edges, is called a **connected component or** simply **component** of *G*.

When *G* has only one component, then *G* is **connected**; otherwise, *G* is **disconnected**.



Disconnected graph with 4 connected components

Algorithm for determining whether graph is connected

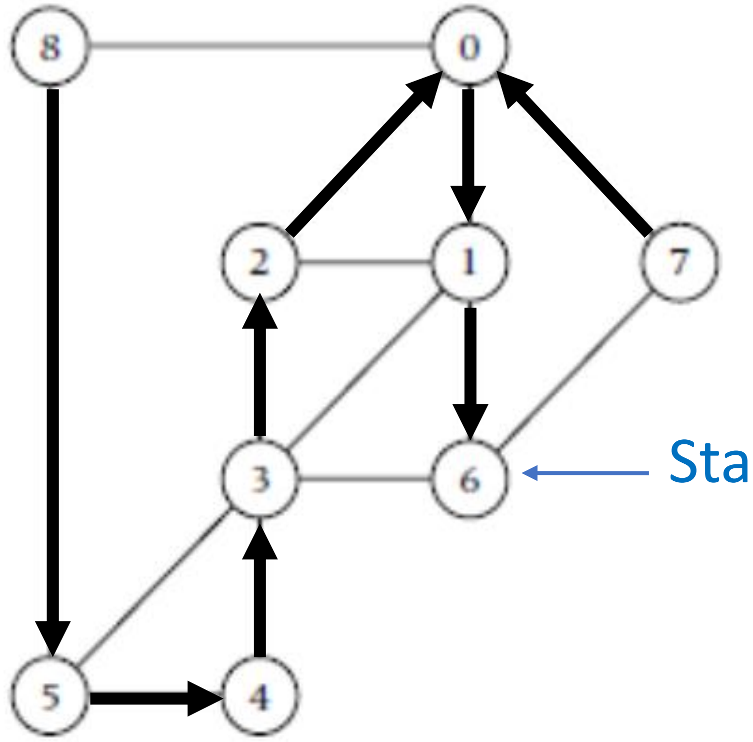
- We can test whether a graph is connected by performing either a Depth-First Search (DFS) or Breadth-First Search (BFS) starting from any vertex of the graph.
- These search techniques are covered in detail an algorithms course, i.e., EECE 4040 and CS 4071.
- Here we introduce DFS.

Depth-First Search (DFS)

```
procedure $DFS(G,v)$ recursive
Input:   $G$ (a graph with $n$ vertices and $m$ edges)
         $v$ (a vertex where search begins)
Output:  the depth-first search of $G$ with starting vertex $v$
    $Mark[v] \leftarrow 1$  // mark $v$ as visited
    call $Visit(v)$
    for each vertex $u$ adjacent to $v$ do
        if $Mark[u] = 0$ then call $DFS(G,u)$ endif
    endfor
end $DFS$
```

Action of DFS for Sample Graph and Initial Vertex

G =



Starting at vertex *v* = 6

DFS tree (obtained by keep parent pointer from *u* to *v*) shown in bold.

DFS order (order in which vertices are visited): 6, 1, 0, 2, 3, 4, 5, 8, 7

Computing Connected Components

The connected components can be computed using Depth-First Traversal (DFT) or Breath-First Traversal (BFT).

DFT and Computing the Connected Components

- The idea behind DFT is to scan all the vertices and if a vertex *v* is unvisited perform a DFS at *v* outputting all the vertices that are visited.
- These visited vertices are the vertex set of the connected component containing *v*.
- Once we have the vertex set we can obtain the component simply by adding all incident edges.

Pseudocode for Depth-First Traversal

```
procedure DFT(G)
Input:        G (a graph with n vertices and m edges)
Output:  Depth-first traversal
      for v ← 0 to n – 1 do
        if Mark[v] = 0 then
              DFS(G,v)
          endif
      endfor
end DFT
```

Coloring Graphs

A **proper** vertex *k*-coloring of a graph is a coloring of the vertices using *k* colors so that there are no monochromatic edges, i.e., both ends of each edge are colored differently.

Finding a proper *k*-coloring is hard except for *k* = 2

Is we mentioned in a previous lecture that finding a proper *k*-coloring is hard even for *k* = 3.

PSN. Describe an algorithm for finding a proper 2-coloring if it exists. Note that this is equivalent to determining whether a graph is bipartite.

Characterization of Bipartite Graphs

A graph is bipartite iff it does not contain an odd cycle.

Equivalently, a graph can be properly 2-colored iff it does not contain an odd cycle.

Proof

Suppose a graph *G* is bipartite. Properly color *G* using colors 0 and 1. Then every cycle is alternately colored 0 and 1, so has even length.

Conversely, suppose G contains no odd cycles. Without loss of generality we can assume G is connected. Choose any vertex u and color the vertex v

$$d(u,v) \mod 2$$

where $d(u,v)$ is the distance between *u* and *v*.

Assume this 2-coloring is not proper. Then there is a monochromatic edge $\{x,y\}$, i.e., x and y are colored the same. But this implies that $d(u,x)+d(u,y)$ is even. It follows that the closed walk consisting of a shortest path from u to x followed by the edge $\{x,y\}$ followed by a shortest path from y to u has odd length. But this walk must contain a cycle of odd length, which contradicts the assumption the G contains no odd cycles. Thus, the 2-coloring is proper.

4-Color Theorem

A famous theorem, which was unsolved more than a century, is known as the 4-color theorem, which states that a planar graph can be properly vertex 4-colored.

It was formally conjectured by Guthrie in 1856 and proved by Appel and Haken in 1970. They use a computer program to evaluate hundreds of different cases.

It is amazing that a planar graph on *n* vertices can be properly vertex colors using only 4 colors. A general graph may require up to *n* colors to properly color the vertices.

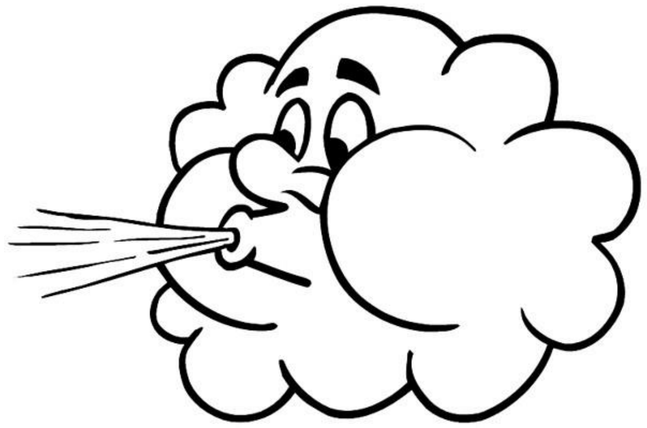
PSN. Which graph on *n* vertices requires *n* colors to properly color?

Application of Proper Coloring

- Suppose we have a set of wireless devices for which we would like to assign channels, so their broadcasts don’t interfere with each other.
- Construct a graph with the devices as vertices
- Join two vertices with an edge whenever that are close enough to interfere with each other.
- A proper *k*-coloring gives an assignment of *k* channels so there is no interference.

What color is the wind?

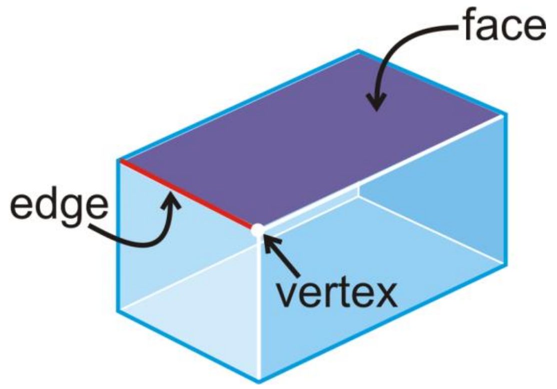
Blew



Planar Graphs

Reading:

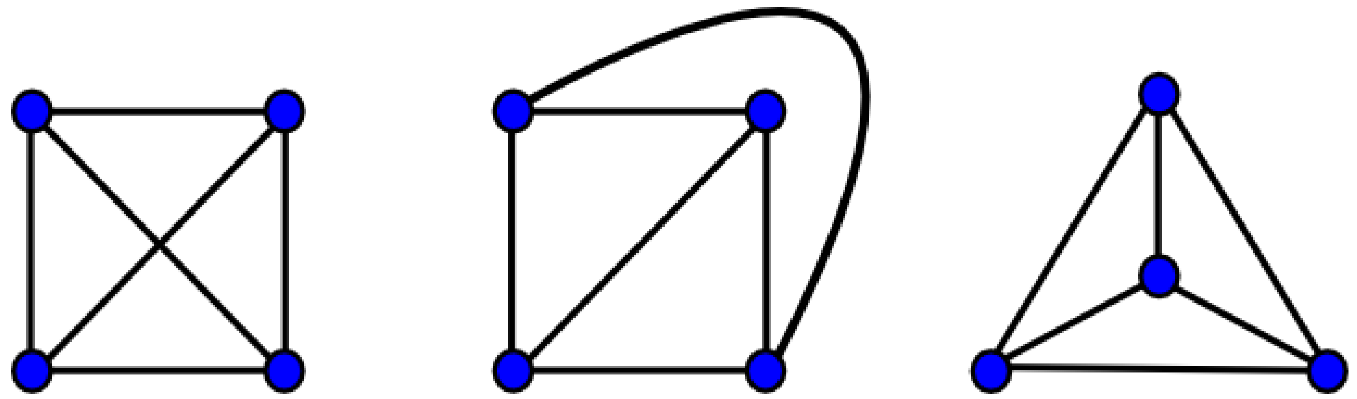
Supplementary Notes
Intro to Graph Theory
for definition of planar graphs and
Euler’s Polyhedron Theorem



Supplement Notes Planar Graphs for Kuratowski’s characterization of nonplanar graphs and the 4-Color Theorem

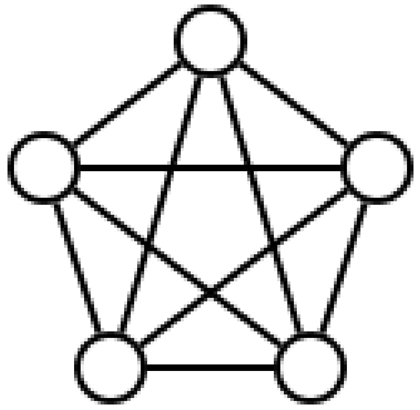
Planar Graphs

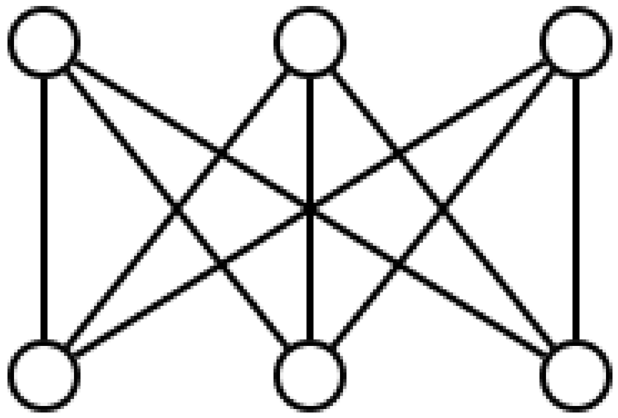
- A fundamental property of graphs is whether or not they can be drawn, i.e., **embedded** in the **plane**, or equivalently on the **sphere**, **without crossing edges**.
- A graph *G* that can be embedded in the plane is called **planar**, otherwise *G* is called **nonplanar**. The regions determined by an embedding of a graph in the plane are called **faces***.*
- We will denote the set of faces by *F* and the number of faces by *f.*
- There are many ways to embed a planar graph in the plane.



Nonplanar Graphs

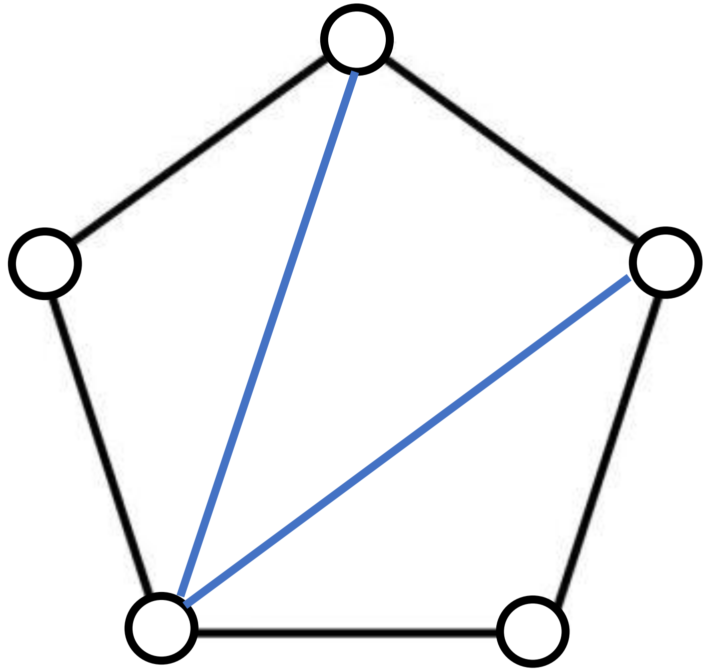
Not all graphs are planar. For example, the graphs K_5 and $K_{3,3}$ are both nonplanar graphs.





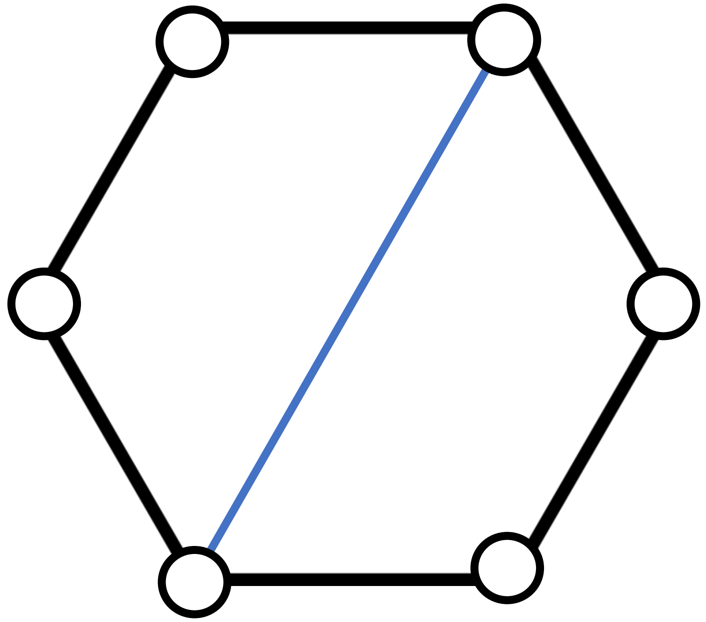
No matter how you draw these graphs in the plane at least two edges will cross.

K_5



At most two edges can be put inside 5-cycle without crossings. Similarly at most two edges can be put outside. Total of only 9 edges. Putting a 10th edge in will result in crossing edges

$K_{3,3}$



Only one edge can be put inside joining opposite ends of 6-cycle. Similarly, only one edge can be put on outside. Total of 8 edges. Putting 9th edge in will result in crossing edges.

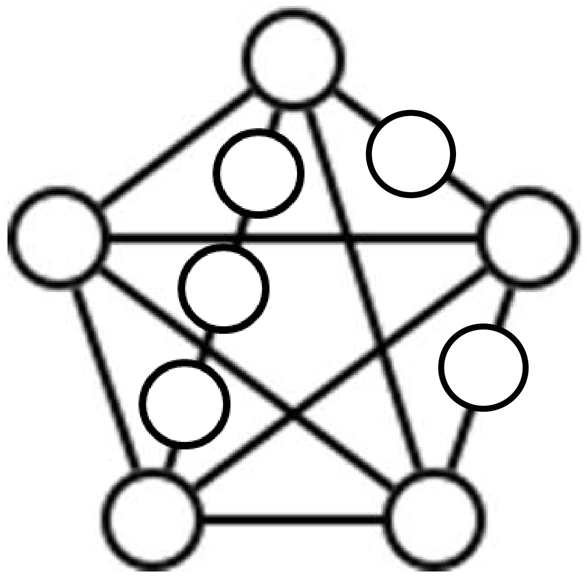
Subdivision of a graph

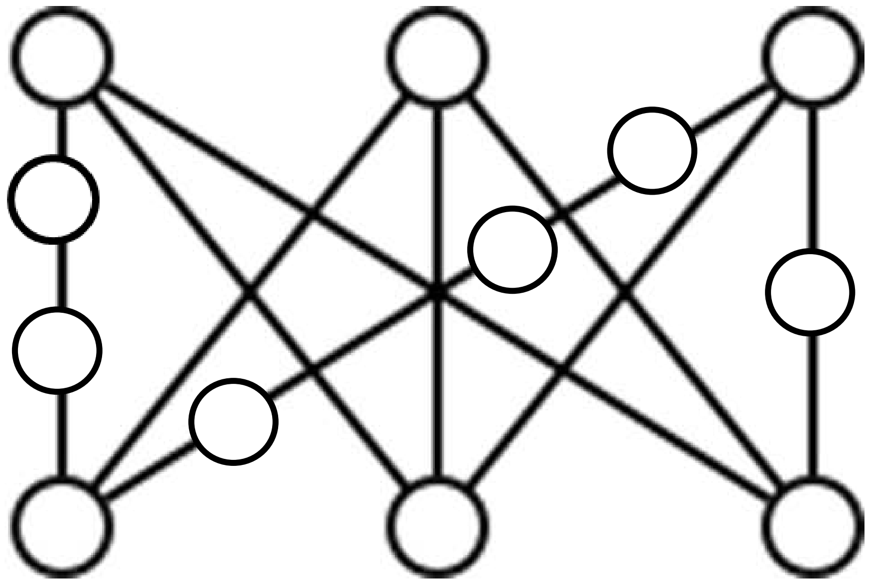
A **subdivision** *S* of *G* is a graph obtained from *G* by replacing each edge *e* with a path joining the same two vertices as *e* (subdividing the edge *e*).

G is a subdivision of itself (replace each edge with a path of length 1).

If *G* is nonplanar, then every subdivision of *G* is nonplanar.

If *G* contains a nonplanar graph, then *G* is nonplanar.



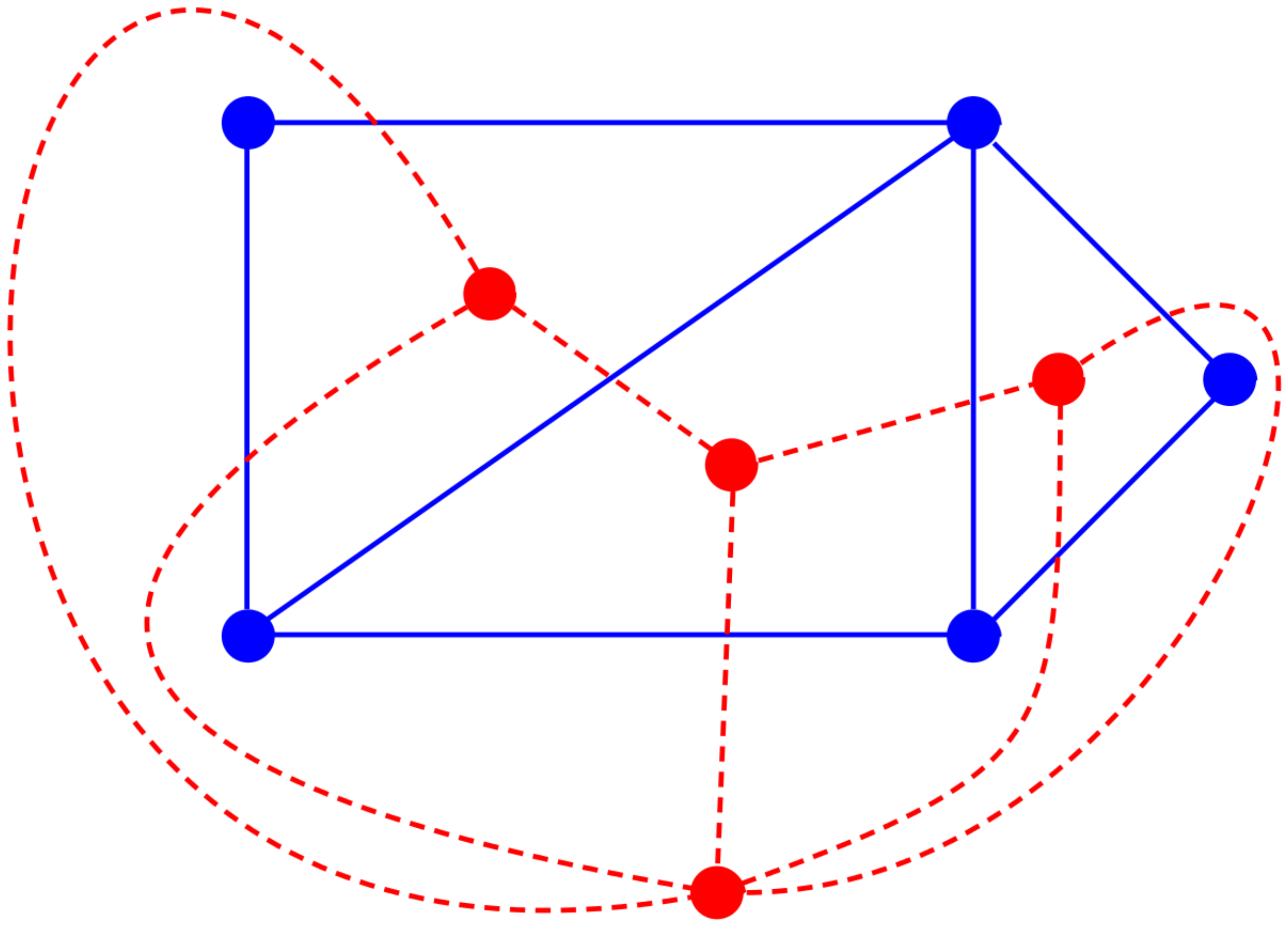


Characterization of non-planar graphs

Clearly, if a graph contains a subgraph that is isomorphic to a subdivision of K_5 or $K_{3,3}$ it is nonplanar. Surprisingly the converse it true. This is a famous result of Kuratowski.

Kuratowski's Theorem. A graph G is nonplanar iff it contains a subgraph that is isomorphic to a subdivision of K_5 or $K_{3,3}$.

Dual Graphs



Euler’s Degree Formulas for faces

The degree of a face is the number of edges incident with the face, i.e., the number of edges on its boundary. Using the dual graph Euler’s degree formula for vertices translates to a degree formula faces:

$$\sum_{g \in F} \deg(g) = 2m$$

Corollary. If *G* is face *s*-regular, then

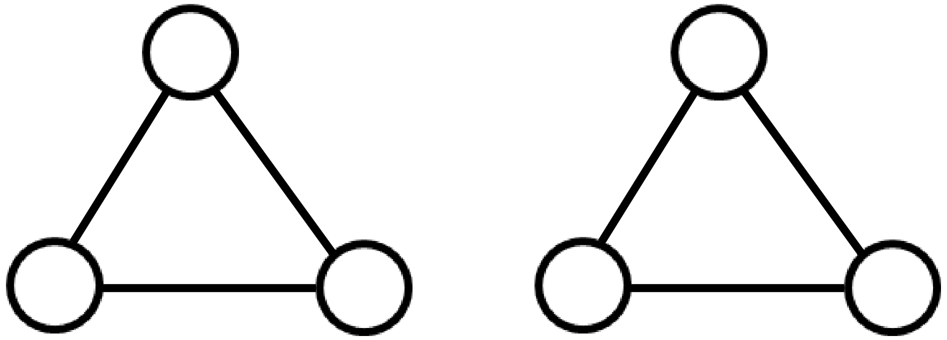
$$m = \frac{sn}{2}$$

Euler's Polyhedron Formula

Let G be a connected planar graph with n vertices, m edges and f faces. Then,

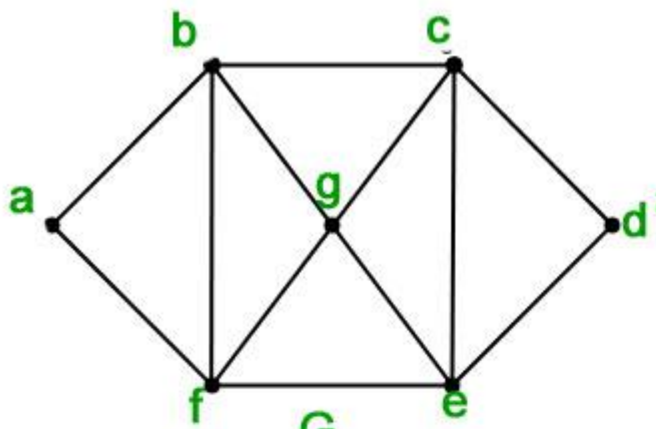
$$n - m + f = 2$$

Note the condition that *G* is connected is necessary. Otherwise, the following disconnected planar graph would be a counterexample.

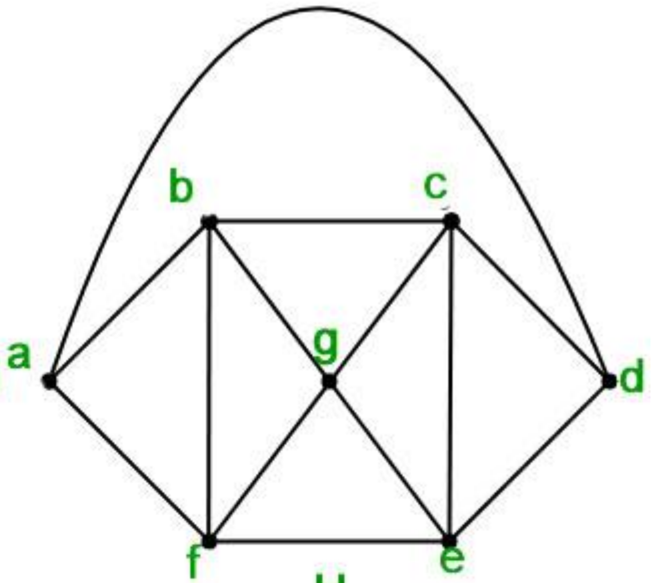


$$n = 6, m = 6, f = 3 \Rightarrow n - m + f = 3$$

PSN. Verify Euler’s polyhedron formula for the following graphs.



G



H

Proof of Euler's Polyhedron Formula using Mathematical Induction

We will perform the induction on the number of edges m.

Basis Step. Since the planar graph must be connected, the fewest edges occurs when the planar graph is a tree in which case

$$m = n - 1$$

If the planar graph is a tree there is only one face, i.e., $f = 1$. Thus, we have

$$n - m + f = n - (n - 1) + 1 = 2.$$

This verifies the basis step.

Induction Step

- Assume true for all connected planar graph having $m = k$ edges.
- Consider **any** connected planar graph *G* having $m = k + 1$ edges.
- Let T be any spanning tree of *G*.
- Since G is not a tree it must contain an edge e not in T.
- Delete edge e to obtain the graph *G’*
- *G’* has k edges so we can apply the Induction Hypothesis to obtain

$$\begin{aligned} 2 &= n(G') - m(G') + f(G') \\ &= n(G) - (m(G) - 1) + (f(G) - 1) \\ &= n(G) - m(G) + f(G) \end{aligned}$$

Q.E.D

Characterization of Regular Polyhedrons

A polyhedron is planar graph that is vertex *r*-regular and face *s*-regular.

By Euler's degree formulas for vertices and faces respectively, we have

$$n = \frac{2m}{r}, \quad f = \frac{2m}{s}.$$

By Euler's Polyhedron Formula

$$n - m + f = 2$$

Substituting we obtain

$$\frac{2m}{r} - m + \frac{2m}{s} = 2$$

Solving for m we have

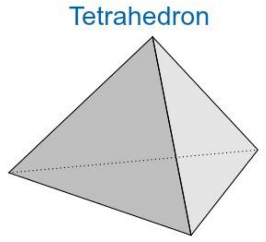
$$m = \frac{2}{\frac{2}{r} + \frac{2}{s} - 1}$$

$$m = \frac{2}{\frac{2}{r} + \frac{2}{s} - 1}$$

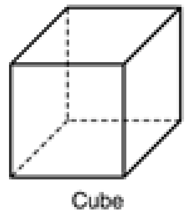
Now $r \geq 3$ and $s \geq 3$. It follows that either $r = 3$ or $s = 3$, because if they were both at least 4, the denominator would be smaller than or equal to 0. Further, both r and s must be no greater than 5, otherwise the denominator would be smaller than or equal to 0. Summarizing

$$3 \leq r \leq 5 \text{ and } 3 \leq s \leq 5 \text{ and either } r = 3 \text{ or } s = 3.$$

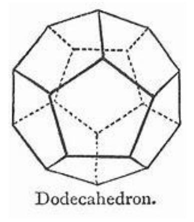
Case $r = 3, s = 3 \Rightarrow m = \dfrac{2}{\frac{2}{r} + \frac{2}{s} - 1} = 6$



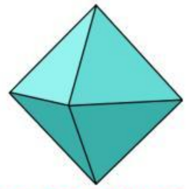
Case $r = 3, s = 4 \Rightarrow m = \dfrac{2}{\frac{2}{r} + \frac{2}{s} - 1} = 12$



Case $r = 3, s = 5 \Rightarrow m = \dfrac{2}{\frac{2}{r} + \frac{2}{s} - 1} = 30$

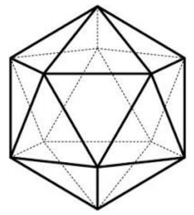


Case $r = 4, s = 3 \Rightarrow m = \dfrac{2}{\frac{2}{r} + \frac{2}{s} - 1} = 12$



octahedron

Case $r = 5, s = 3 \Rightarrow m = \dfrac{2}{\frac{2}{r} + \frac{2}{s} - 1} = 30$



icosohedron

Average degree of a vertex in a planar graph

Note that every face must have degree at least 3, otherwise the graph would be a multigraph with two edges joining the same pair of vertices. Thus, by Euler's degree formula for faces, we have

$$2m = \sum_{g \in F} \deg(g) \geq \sum_{g \in F} 3 = 3f$$

Thus, $f \leq \frac{2}{3}m$. Substituting in Euler's Polyhedron Formula

$$2 = n - m + f \leq n - m + \frac{2}{3}m$$

$$\Rightarrow m \leq 3n - 6.$$

Using the formula we derived using Euler's degree formula for the average degree α of a vertex and substituting we obtain

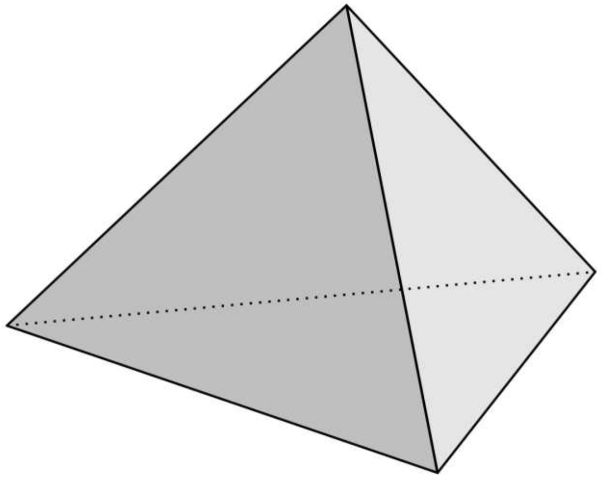
$$\alpha = \frac{2m}{n} \leq \frac{2 \times (3n - 6)}{n} = 6 - \frac{12}{n} < 6$$

- We showed that the average degree of a vertex in a planar graph is strictly less than 6.
- It follows that there must exist a vertex of degree 5 or smaller.

PSN. Using the above result, apply mathematical induction to show every planar graph can be properly 6-colored.

Why did the polyhedron go to jail?

For running a pyramid scheme.



Spanning Trees, Eulerian Circuits

Textbook Reading

Chapter 6, Section 6.11 pp. 374-377 (Spanning Trees)
Section 6.8, pp. 361-364 (Eulerian Circuit)

Definition of a Spanning Tree

Let G be a connected graph with n vertices and m edges.

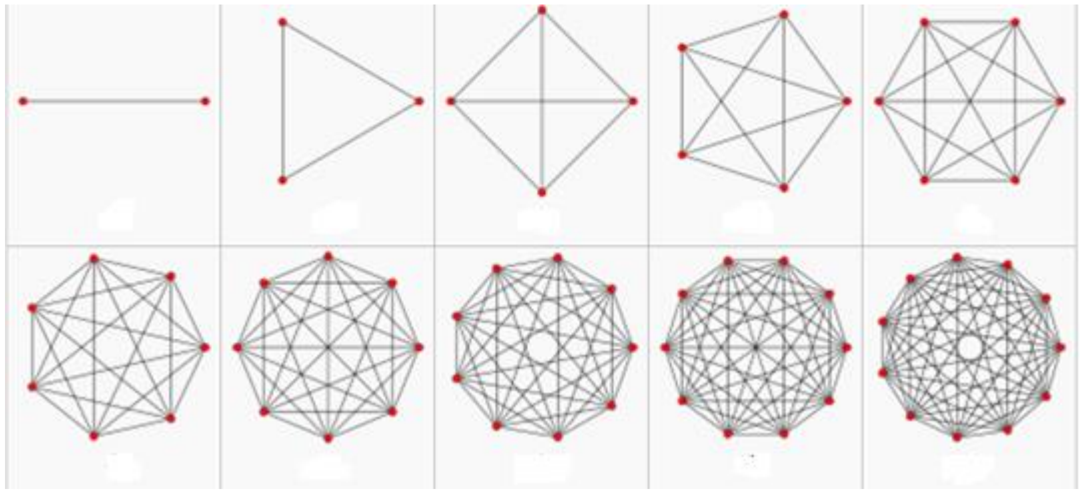
A **spanning tree** is a tree of G that contains, i.e., spans, all the vertices.

The number of edges of a spanning tree is $n - 1$.

Enumerating Spanning Trees

Clearly, the complete graph K_n has the most spanning trees for a graph on n vertices

1 3 16 125 1296



16807 262144 4782969 100000000 2357947691

Can you guess the formula for the number of spanning trees of K_n?

Cayley’s Theorem

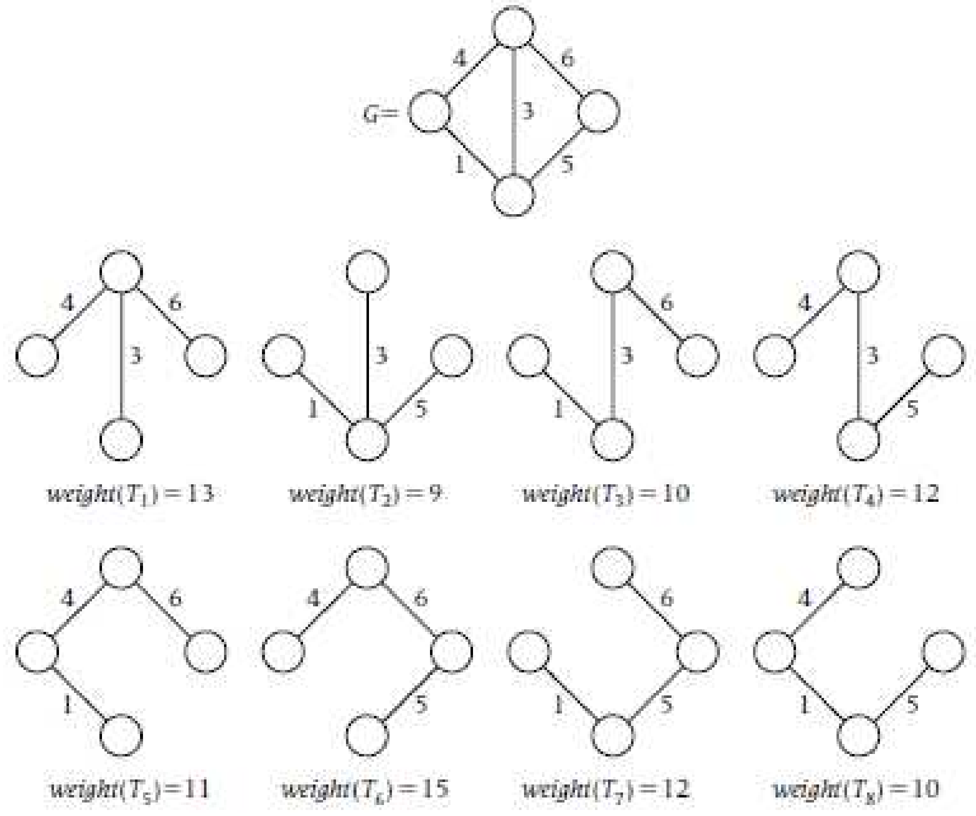
By a theorem of Caley the number of spanning trees of K_n, the complete graph on n vertices is

$$n^{n-2}.$$

Minimum Spanning Tree

- Now associated a positive real weight $w(e)$ with each edge $e \in E$.
- The **weight** of a spanning tree of T, denoted *w*(*T*), is the sum of the *w*-weights over all its edges.
- If *T* has minimum weight over all spanning trees of *G*, then we call *T* a **minimum spanning tree (MST)**.

Minimum Spanning tree for Sample Graph *G*



All 8 spanning trees of sample graph G are shown above. The minimum spanning tree has weight 9.

Kruskal's Algorithm

Input: Connected graph $G = (V,E)$, weighting w of the edges

Output: Minimum Spanning Tree T, i.e., weight(T) is minimum

Design Strategy employed: the greedy method

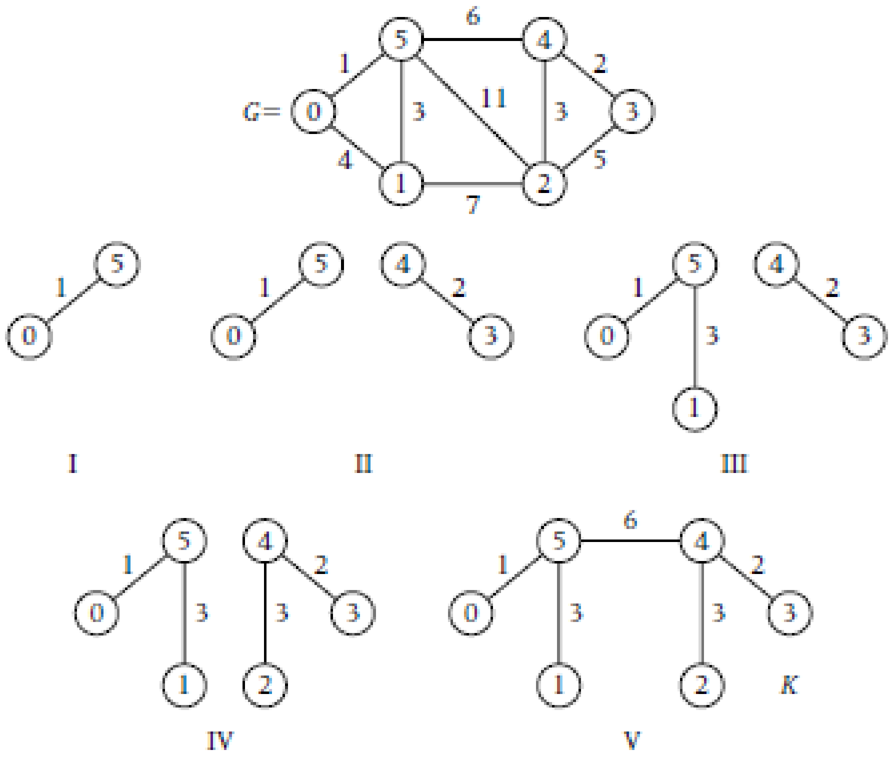
Notation: As usual we let $n = |V|$ (order of G) and $m = |E|$ (size of G)

Compute a sequence of n forests $F_0, F_1, \ldots, F_{n-1}$, where **$F_0$ is the empty forest, i.e., consists of a set of *n* isolated nodes** and F_i is obtained from F_{i-1} by adding a single edge e_i, denoted

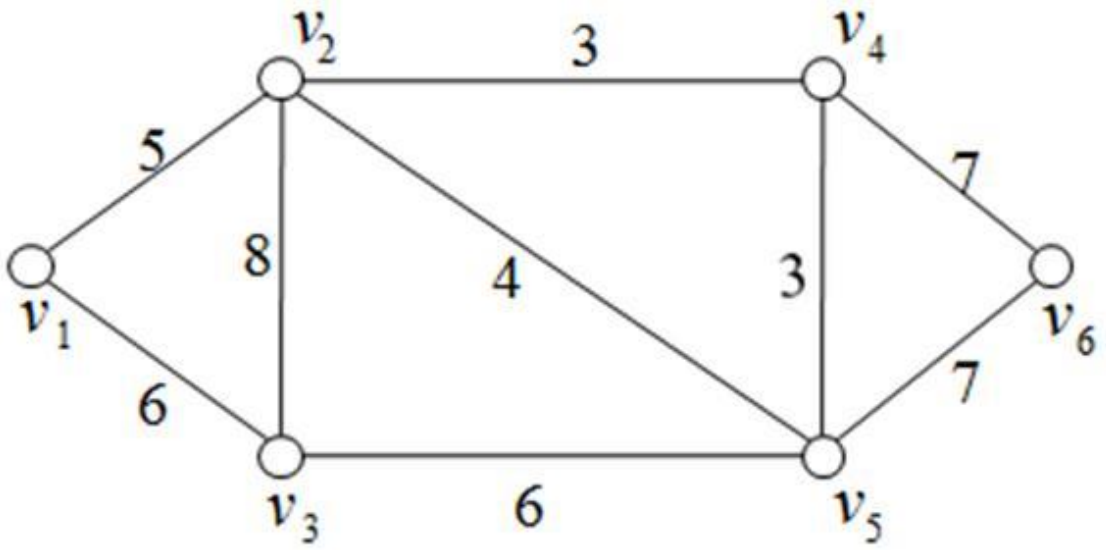
$$F_i = F_{i-1} + e_i\ ,\ i = 1, 2, \ldots, n - 1,$$

where e_i is chosen so that it has **minimum weight** among all the edges not previously chosen and **doesn't form a cycle** when added to F_{i-1}.

Action of Kruskal's for sample graph *G*

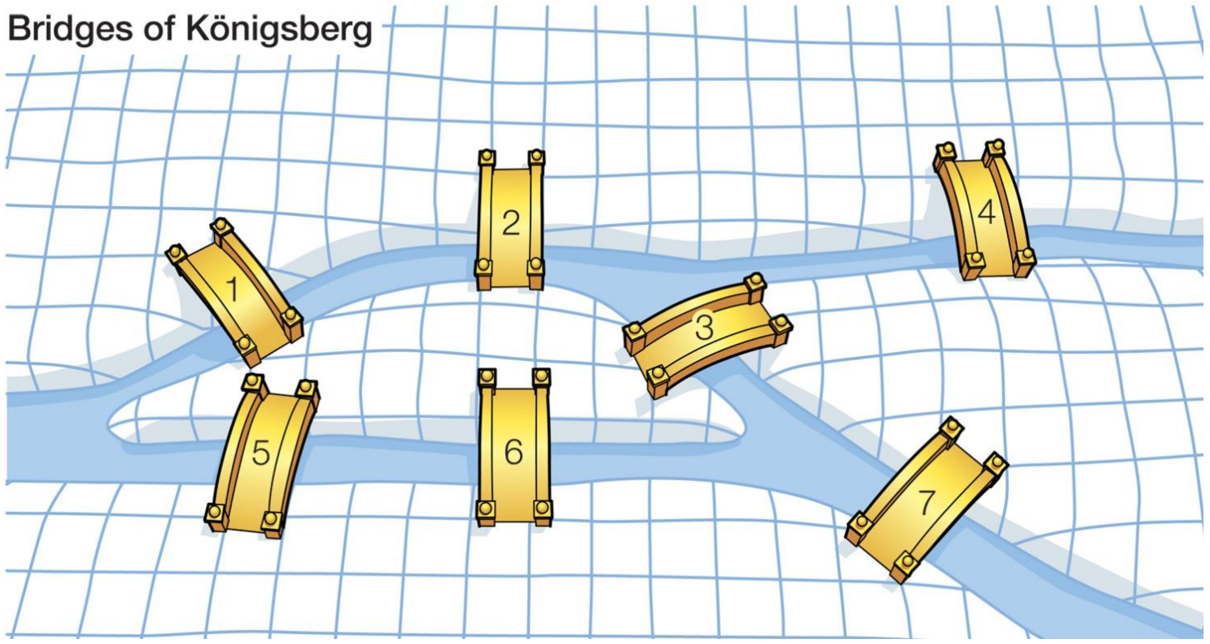


PSN. Find a MST in the following weighted graph.



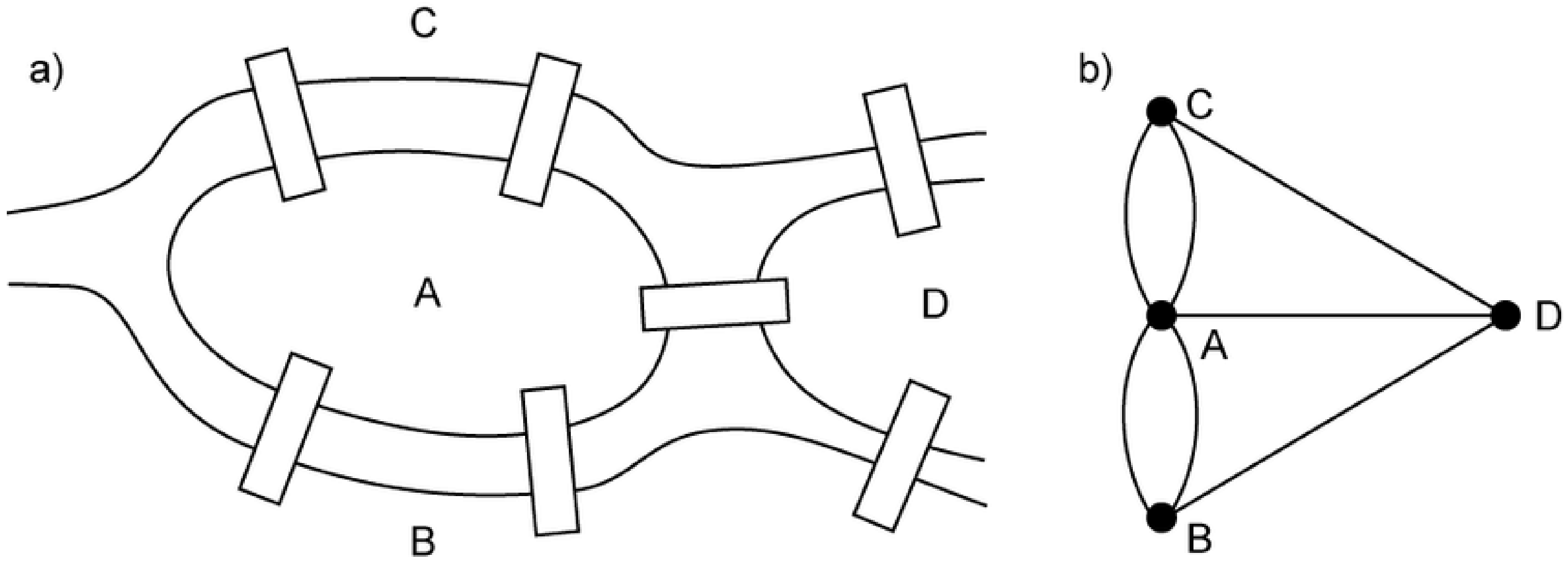
Konigsberg Bridge Problem

- Is it possible to take a stroll around Königsberg (now called Kaliningrad) crossing every bridge once and return to where one started.



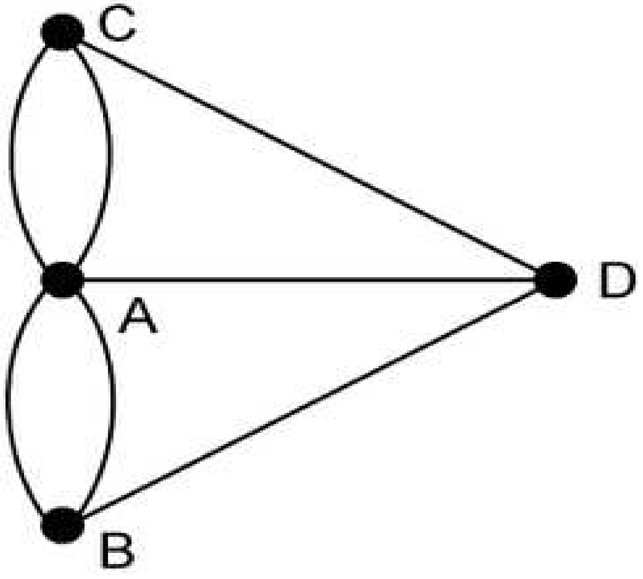
Graph Modelling

- The Konigsberg Bridge Problem was solved by the famous mathematician Euler, by modelling with a graph.



The solution is to find an Eulerian Tour or Eulerian Circuit, i.e., a circuit that contains every edge.

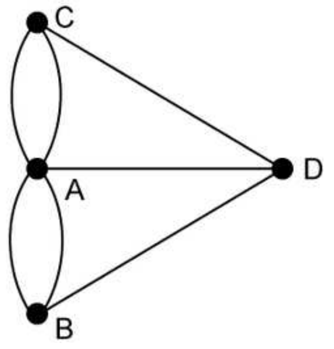
We model with a **multigraph**. There are two edges (a multiedge with multiplicity 2) joining A and B and A and C.



Eulerian Circuit

Theorem 1. A multigraph G contains an Eulerian circuit iff it is connected and every vertex has even degree.

All the vertices of the multigraph corresponding to the Konigsberg Bridge Problem have odd degree. Therefore it doesn’t contain a Eulerian Circuit.



Proof of Theorem 1

- If G is disconnected then clearly if doesn’t contain an Eulerian circuit.
- When ever an Eulerian circuit enters a vertex, it must leave the vertex.
- Therefore, the number of edges used to enter each vertex v equals the number of edges used to leave v.
- Thus, the number of edges of the circuit incident with v is even.
- But, an Eulerian circuit contains each edge.
- It follows that every vertex of G has even degree.

Proof of Theorem 1 cont'd

- Conversely, suppose G is connected and every vertex has even degree.
- We use Proof by Contradiction to show G contains an Eulerian circuit.
- Suppose to the contrary that G does not contain an Eulerian circuit.
- Let C *be a* **longest** circuit*, i.e.,* $C = u_0 u_1 \cdots u_j u_0$ where j is maximum.

Proof cont'd

- Since G is connected and C is not an Eulerian circuit, then, there must exist a vertex u_i of C such that u_i is adjacent to some vertex v_1 not in C.
- Mark the vertices of C as visited.
- Since every vertex of G is even, there are an even number of unvisited vertices in the neighborhood of every vertex.
- Grow a trail starting with the edge $u_i v_1$and keep choosing as the next vertex an unvisited vertex.
- Eventually, the trail must return to vertex u_i.
- Denote this closed trail, i.e., circuit, by $C' = u_i v_1 \cdots v_k u_i$
- Construct the circuit C'' by splicing C at vertex u_i and inserting the circuit C', i.e., $C'' = u_0 u_1 \cdots\ u_i v_1 \cdots v_k u_i \cdots u_j u_0$.
- C'' is a longer than C, contradicting our assumption that C is a longest circuit.

Q.E.D.

Eulerian Trail

Corollary. A multigraph contains an Eulerian trail iff it is connected and exactly two vertices have odd degree.

PSN. Prove the corollary using Theorem 1.

What did the beaver say to the tree?

It’s been nice gnawing you!

Hypercubes and Hamiltonian Cycles

Reading:

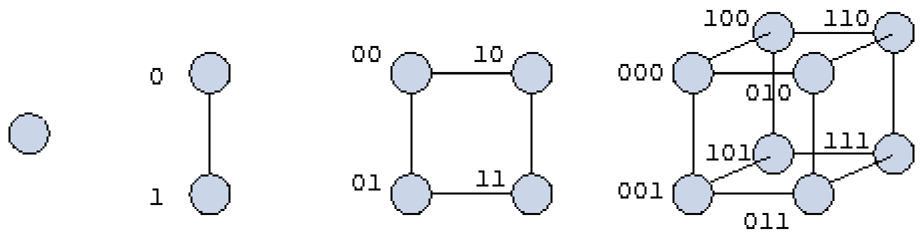
Supplemental Notes on Hypercubes and Hamiltonian Cycles

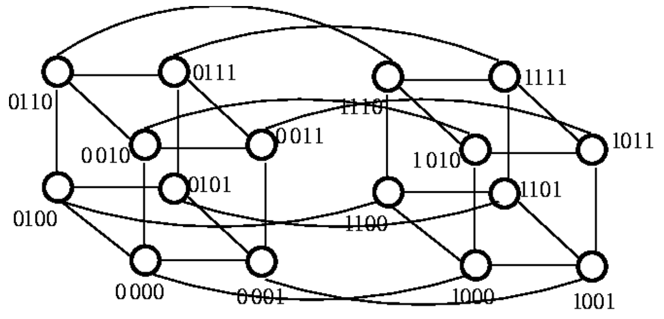
Hypercube Definition

The k-dimensional hypercube H_k has 2^k vertices consisting of the set of all 0/1 k-tuples, i.e.,

$$V(H_k) = \{(x_1, . . . , x_k) \mid x_i \in \{0,1\}, i = 1, . . . , k.$$

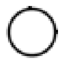
Two vertices in $V(H_k)$ are joined with an edge of H_k whenever they differed in exactly one component.

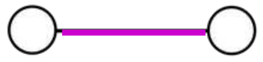


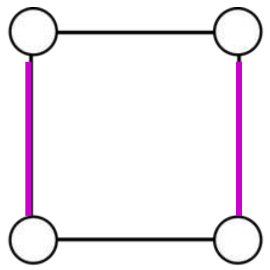


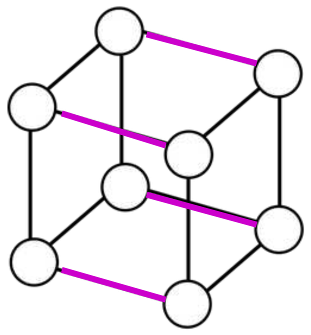
Recursive Construction of a Hypercube

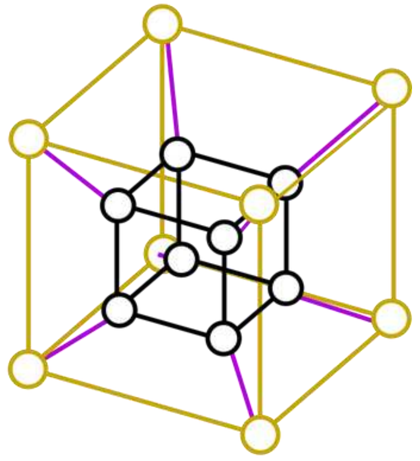
The hypercube H_k of dimension k is obtained from the hypercube H_{k-1} of dimension k – 1 by taking two isomorphic copies of H_{k-1} and joining corresponding vertices with a matching.


0d hypercube


1d hypercube


2d hypercube


3d hypercube


4d hypercube

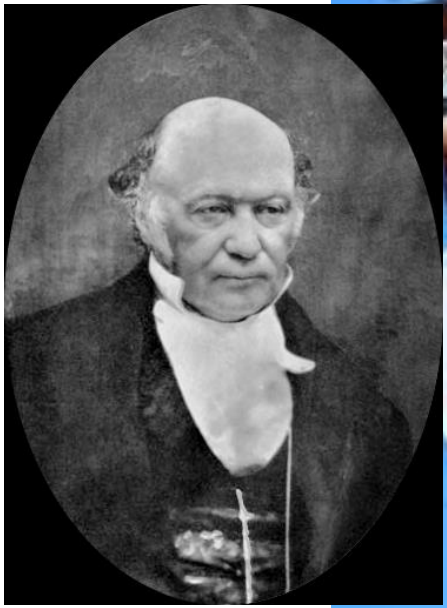
PSN.

a) Obtain a formula for the number of edges of H_k.

b) Obtain a formula for the diameter of H_k.

Hamiltonian Cycle

Sir William Rowen Hamilton's Icosian Game



Goal of Icosian Game

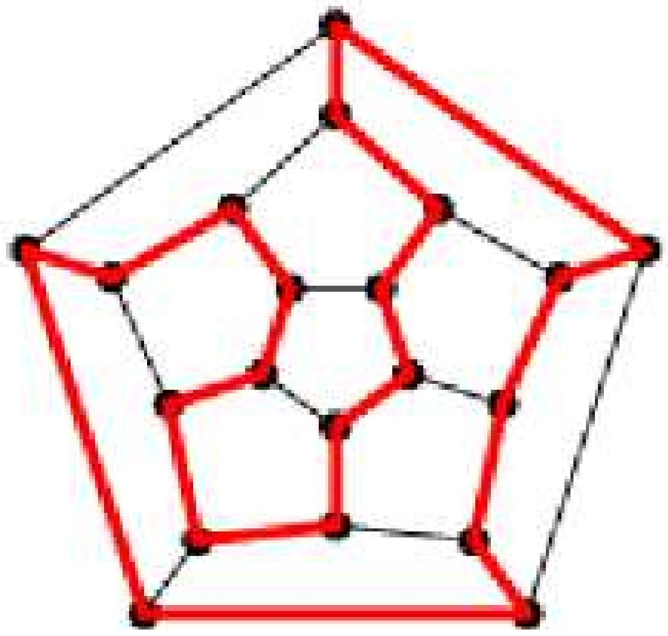
Vertices of the icosahedron represent cities. The goal is to perform a tour of the 20 cities and return to the starting vertex, following an edge of the icosahedron to move between two cities.

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:



Hamiltonian cycles

A Hamiltonian cycle is a cycle that contains all the vertices. The problem of determining whether a graph has a Hamiltonian cycle is NP-complete.

Gray Codes

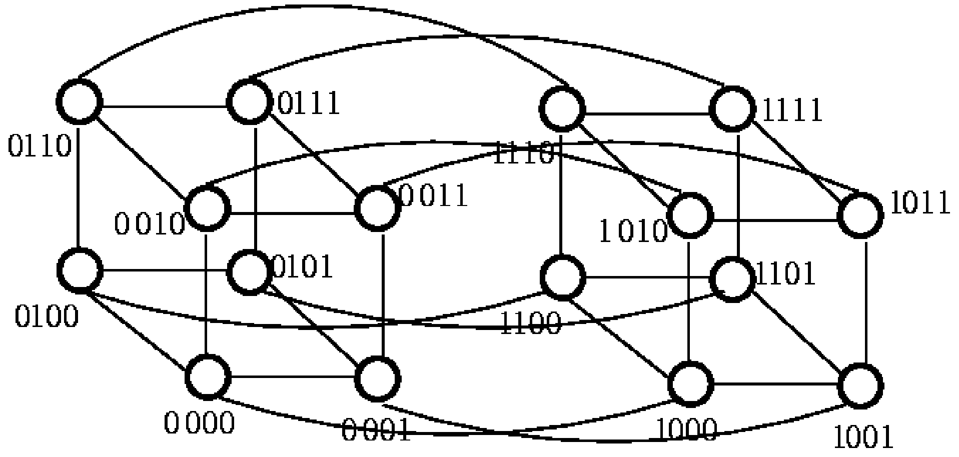
***k*-bit Gray code** is an ordering of *k*-bit strings so that consecutive strings differ in exactly one position.

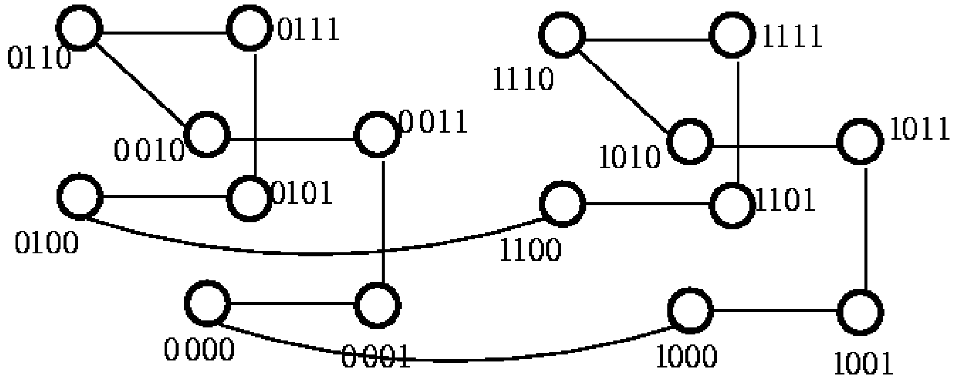
An example of a 4-bit Gray Code:

0000 0001 0011 0010 0110 0111 0101 0100 1100 1101 1111 1110 1010 1011 1001 1000 0000

Gray codes have applications to rotary and optical encoders, Karnaugh maps and error detection.

Gray Codes correspond to Hamiltonian Cycles in Hypercubes



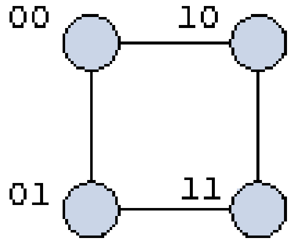


0000 0001 0011 0010 0110 0111 0101 0100 1100 1101 1111 1110 1010 1011 1001 1000 0000

Theorem. The k -dimensional hypercube H_k contains a Hamiltonian cycle for all $k \geq 2$.

Proof by Induction.

Basis Step. H_2 contains a Hamiltonian cycle.



Induction Step

- Assume the Theorem is true for H_k , i.e., H_k contains a Hamiltonian cycle $C = u_0 u_1 \cdots u_{n-1} u_0$, where n is the number of vertices of H_k, i.e., $n = 2^k$.

- Consider the hypercube H_{k+1}. Now H_{k+1} consists of two isomorphic copies H_k. Let $C' = u'_0 u'_1 \cdots u'_{n-1} u'_0$ be the cycle corresponding to C in the second copy.

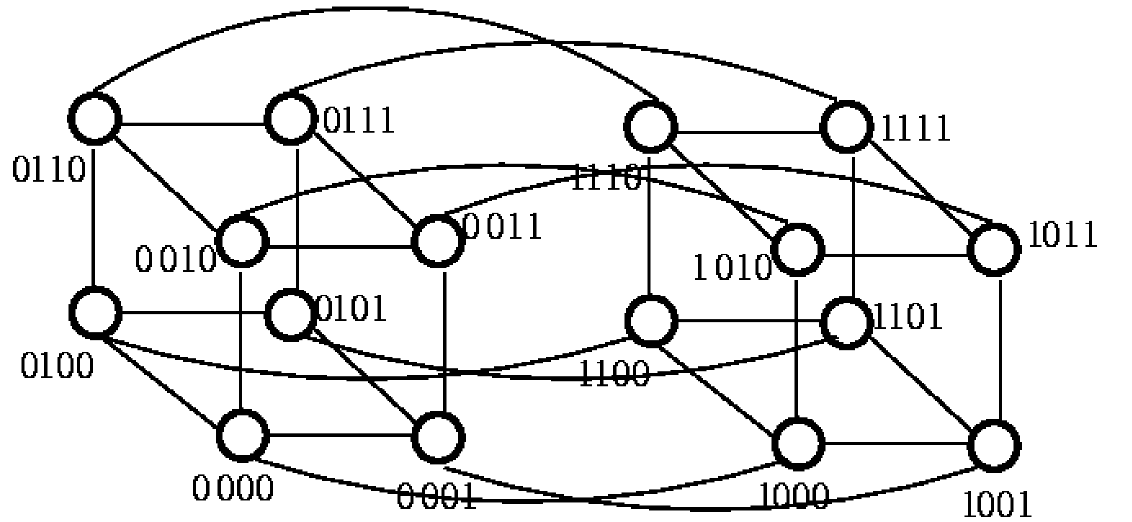
- By the recursive construction there is a matching joining corresponding vertices. In particular there is an edge joining u_0 and u'_0 and an edge joining u_1 and u'_1.

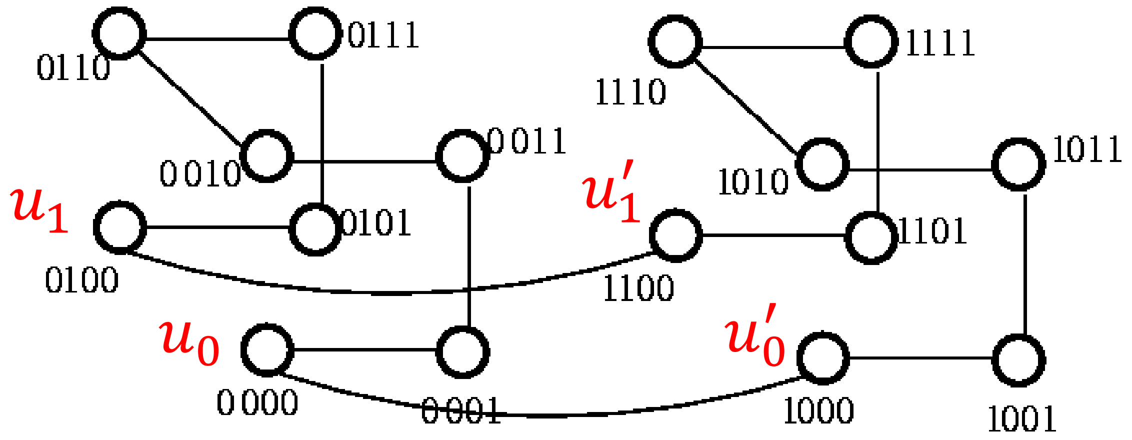
- We construct a Hamiltonian cycle in H_{k+1} as follows:

$$u_1 \cdots u_{n-1} u_0 u'_0 u'_{n-1} u'_{n-2} \cdots u'_1$$

This complete the Induction Step. Q.E.D.

Illustration of Induction Step for $k = 3$





Hamiltonian cycles in Dense Graphs

Theorem 3.2 (Dirac). A graph *G* with $n \geq 3$ vertices having minimum degree at least $\frac{n}{2}$ is Hamiltonian, i.e., contains a Hamiltonian cycle.

PSN

Show that Dirac's Theorem is as strong as possible by giving a counterexample of a graph where every vertex has degree at least $\frac{n}{2} - 1$, but the graph is not Hamiltonian.

a) Find with a counterexample where the graph is not connected for n = 10.

b) Find a counterexample for any n, where n is even.

A generalization of Dirac’s Theorem

Theorem 3.3 (Ore). Let *G* be a graph with $n \geq 3$ vertices such that, for every pair of nonadjacent vertices u and v, $\deg(u) + \deg(v) \geq n$. Then *G* is Hamiltonian.

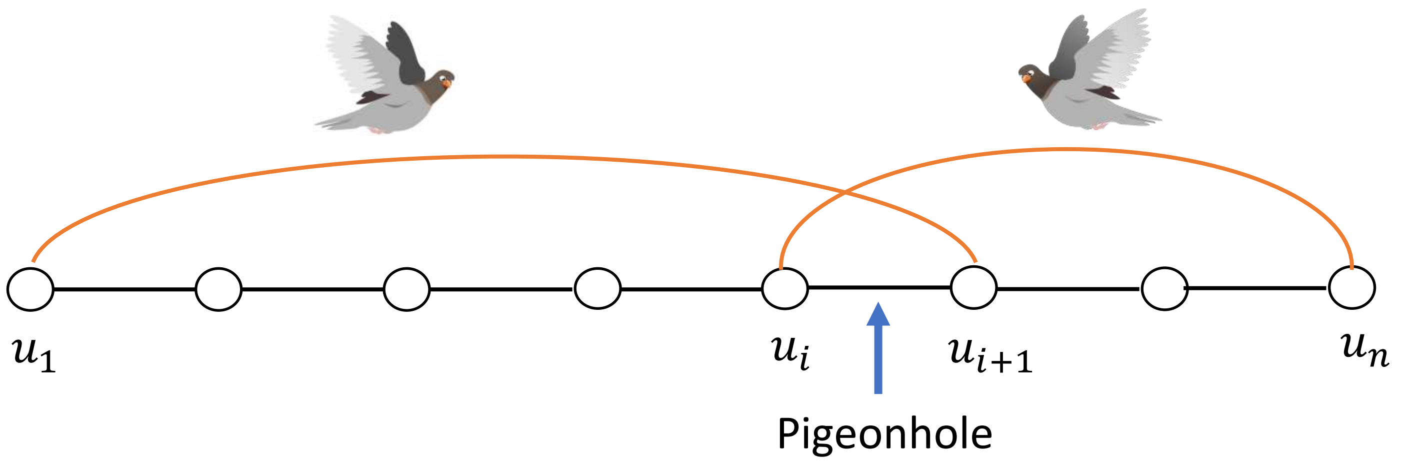
Proof of Ore's Theorem

- Let $G = (V, E)$ be a maximal graph on *n* vertices that satisfies Ore's conditions but which is not Hamiltonian.
- Clearly, *G* is not the complete graph, so that there exists at least one pair *u*, *v* of nonadjacent vertices.
- It follows from the maximality of *G* that the graph *G*+*uv* obtained by adding an edge *uv* contains a Hamiltonian cycle that contains edge *uv*.
- It follows that *G* contains a u-v Hamiltonian path

$$P = u_1 u_2 \ldots u_n \text{ (where } u = u_1, v = u_n).$$

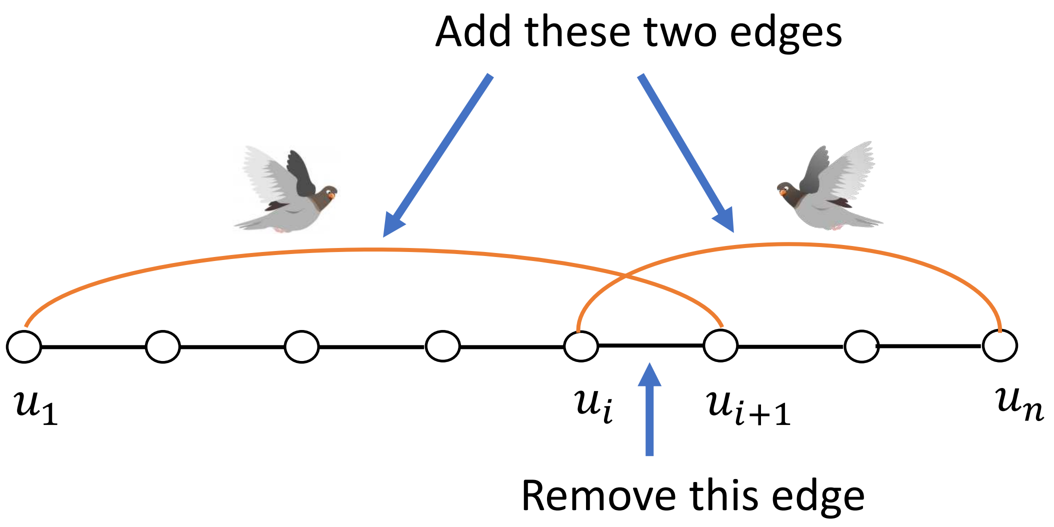
Proof of Ore's Theorem cont'd

- $n-3$ pigeonholes: edges of path $u_2u_3, u_3u_4, \dots, u_{n-2}u_{n-1}$
- Pigeons: edges incident to either u_1 or u_n that are not in the path
- Since $\deg(u)+\deg(v) \geq n$, it follows that there are at least n edges incident to either u_1 or u_n, so there are at least $n-2$ pigeons
- For $u_1u_{i+1} \in E$ have pigeon fly into pigeonhole $u_iu_{i+1}, i = 2, \dots, n-2$.
- For $u_nu_i \in E$ have pigeon fly into pigeonhole $u_iu_{i+1}, i = 2, \dots, n-2$.
- Note that pigeons from u_1 fly into different pigeonholes and pigeons from u_n fly into different pigeonholes.



Conclusion of Proof

- Since there are more pigeons than pigeonholes two pigeons $u_1 u_{i+1}$ and $u_n u_i$ must fly into the same pigeonhole. Thus, $u_1 u_{i+1} \in E$ and $u_n u_i \in E$.



- Construct the Hamiltonian circuit

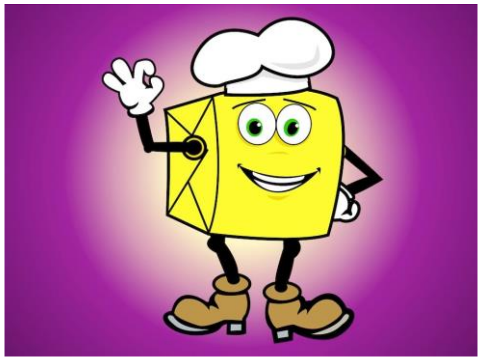
$$u_1 u_2 \ldots u_i u_n u_{n-1} u_{n-2} \ldots u_{i+1} u_1$$

- But this contradicts our assumption that G is not Hamiltonian.

  Q.E.D.

How do you add flavor to your algorithm?

Use a Boolean cube.



Implementation of Graphs and Digraphs

Textbook Reading:

Chapter 6, Section 6.5, pp. 346-347
Section 6.14, pp. 392-394.

Standard Implementations

Let $G = (V,E)$ be a graph, where $V = \{0,1, . . . , n - 1\}$

Two standard implementations of G are

- adjacency matrix implementation
- adjacency lists implementation

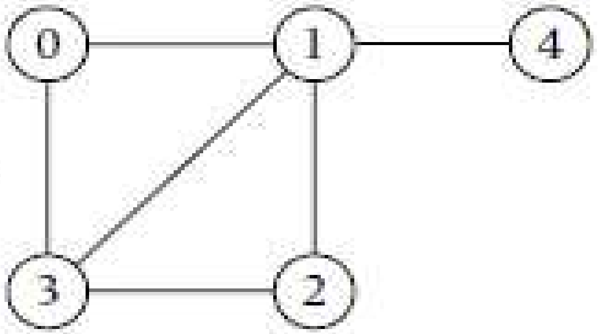
Adjacency Matrix Implementation

The *adjacency matrix* of a graph G is the $n \times n$ symmetric matrix $A = (a_{ij})$ given by

$$a_{ij} = \begin{cases} 1 & \text{vertices } i \text{ and } j \text{ are adjacent in } G. \\ 0 & \text{otherwise,} \end{cases} \qquad i, j \in \{0, \ldots, n-1\}.$$

Sample Graph and its Adjacency Matrix

$G=$



$$A=\begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

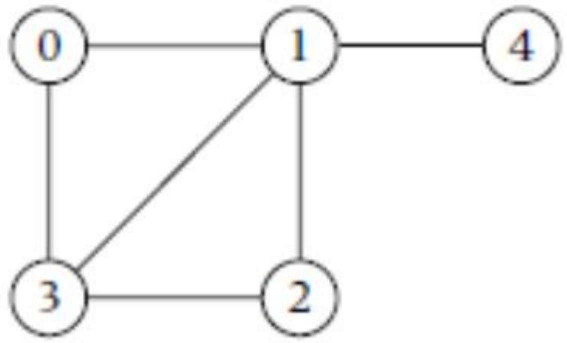
Pros and Cons

- Implementing G using its adjacency matrix makes it easy to perform many standard operations on G, such as adding a new edge or deleting an existing edge.
- The adjacency matrix of G allocates n^2 memory locations no matter how many edges are in the graph.
- Implementing G using its adjacency matrix is inefficient if the graph is **sparse**, i.e., the number of edges of G is small relative to the number of vertices.
- For example, if G is a tree with n vertices, then G has only $n – 1$ edges, which means only n – 1 out of n^2 entries of the matrix are being used, i.e., have the value 1.
- If graph is **dense**, e.g., $m \in \Theta(n^2)$, then the adjacency matrix representation is an efficient way to implement G.

Adjacency Lists Implement

In the adjacency lists implementation we represent the graph with a list of edges the are adjacent to each node.

G =



0 : 1 3
1 : 0 2 3 4
2: 1 3
3: 0 1 2
4: 1

Order Doesn’t Matter

The order that the vertices are listed in the adjacency lists does not matter. Another representation could be

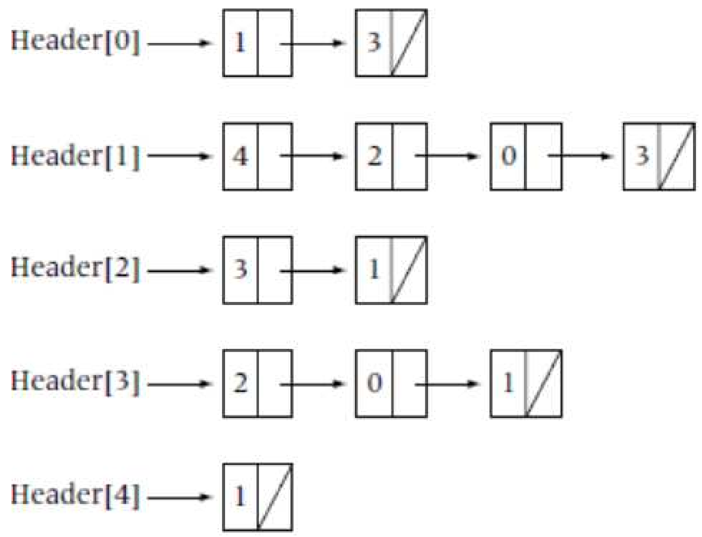
```
0 : 1  3
1 : 4   2  0
2:  1   3
3:  2   0   1
4:  1
```

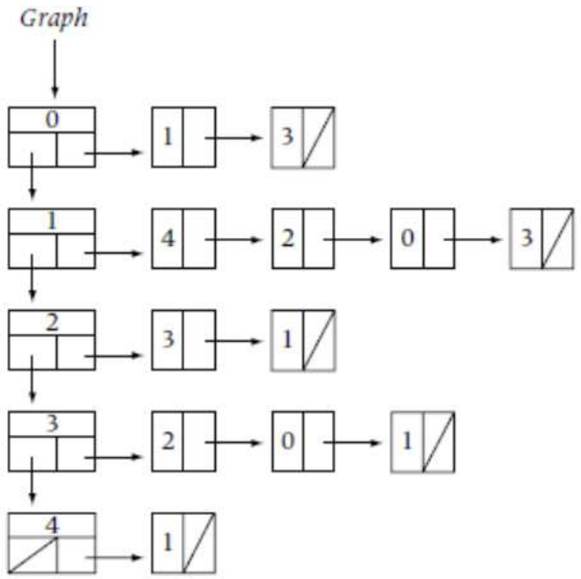
The implementation of a graph for I/O (input-output) simply stores the collection of vertices and the collection of pairs representing the edges of the graph. For example, there are XML format for graphs such as Graph Modelling Language (GML) and GraphML and that stores the graph this way.

When a graph is input the edges, i.e., pair of vertices, are inserted into the adjacency list in the order they are read.

Implementation of lists and graphs

A discussion of the implementation of lists using linked lists is beyond the scope of this course. It is covered in a Data Structures course. Below pictorially shows two implementations of a graph using linked lists, the first with an array of header nodes and the second with a linked list of header nodes.





Adjacency Matrix vs. Adjacency Lists

Adjacency Matrix allows for direct access so is more efficient for adding and deleting edges.

Adjacency Lists is more efficient in terms of storage

If a graph is dense, i.e., close to the complete graph it is often better to use adjacency matrix

If the graph is sparse it is generally more efficient to use adjacency lists

Must use Adjacency Lists for Big Graphs

- For big graph such as the friendship graph for Facebook, i.e., the vertices are users and two users are joined with an edge if there friends, then it is necessary to use the adjacency list implementation.
- Number of users on Facebook is over one billion.
- Therefore size of adjacency matrix would be the square of a billion would be a billion billion or 1,000,000,000,000,000,000
- Much too large to store!
- On the other hand the average number of friends of a user on Facebook is estimated to be about 338.
- It follows from Euler's degree formula that the average degree equals $2m/n$. Therefore we have

$$338 = \frac{2m}{n} \Rightarrow 2m = 338n = 338{,}000{,}000{,}000$$

- The size of the adjacency lists would be twice the number of edges plus the number of vertices or

$$2m + n = 339{,}000{,}000{,}000$$

- 339 billion is not too large to store.

Digraphs

- A digraph $D = (V, E)$ consists of a vertex set V and an (directed) edge set E, where the (directed) edges are (ordered) pairs of vertices.

- Equivalently, E is a relation on the set V, called an **adjacency relation**.

- For $(a, b) \in E$ we call a the **tail** and b the **head**.

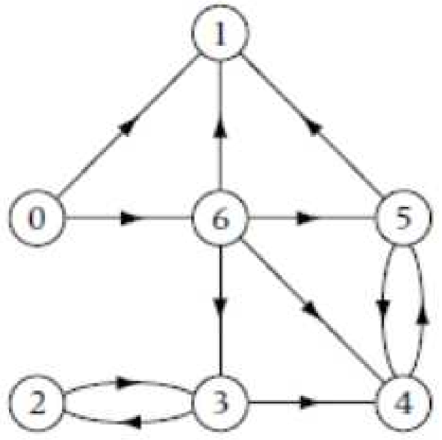
- For convenience (a, b) can be written simply as ab.

Drawing a digraph in the plane

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

$$E = \{(0,1),(0,6),(2,3),(3,2),(3,4),(4,5),(5,1),(5,4),(6,1),(6,3),(6,4),(6,5)\}$$

$D=$



Implementation of Digraphs

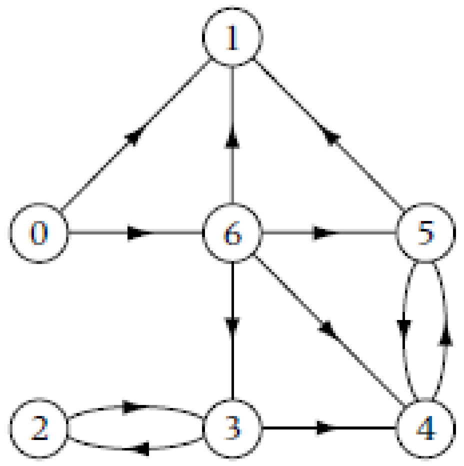
Digraphs are implemented in a similar way to graphs except the edges are ordered.

The *adjacency matrix* of a digraph *D*, whose vertices are labeled 0,1, . . . , *n* – 1, is the *n* × *n* matrix *A* = (a_{ij}) defined by

$$a_{ij} = \begin{cases} 1 & \text{if there is an edge from } i \text{ to } j \\ 0 & \text{otherwise} \end{cases}$$

Adjacency Matrix for Sample Digraph

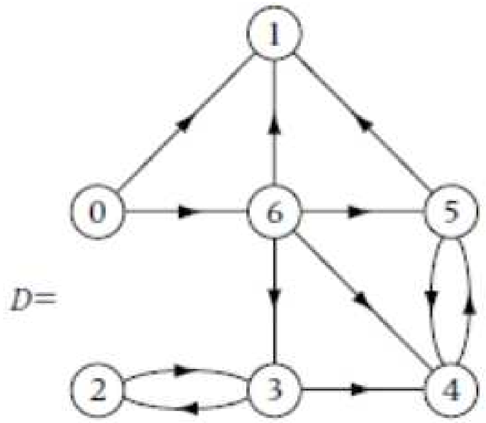
$D=$

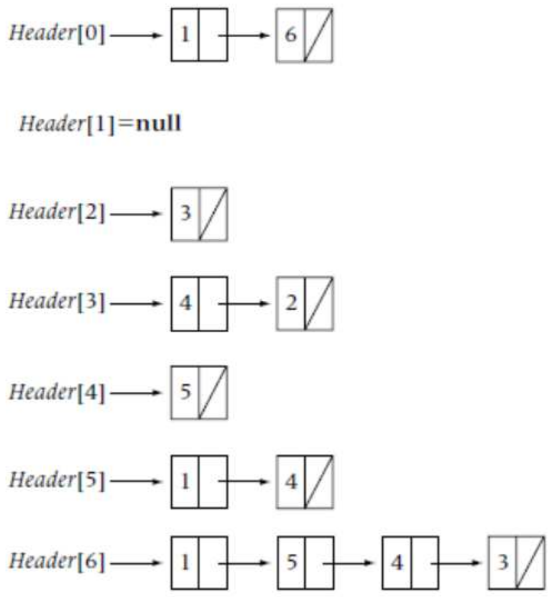


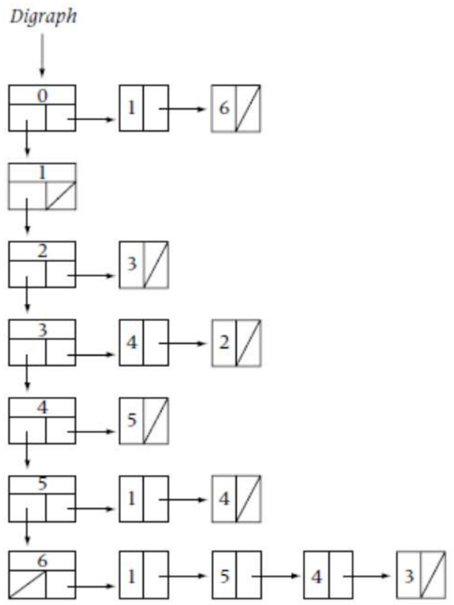
$$A=\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

Adjacency List Implementation of a Digraph

0: 1 6
1:
2: 3
3: 4 2
4: 5
5: 1 4
6: 1 5 4 3



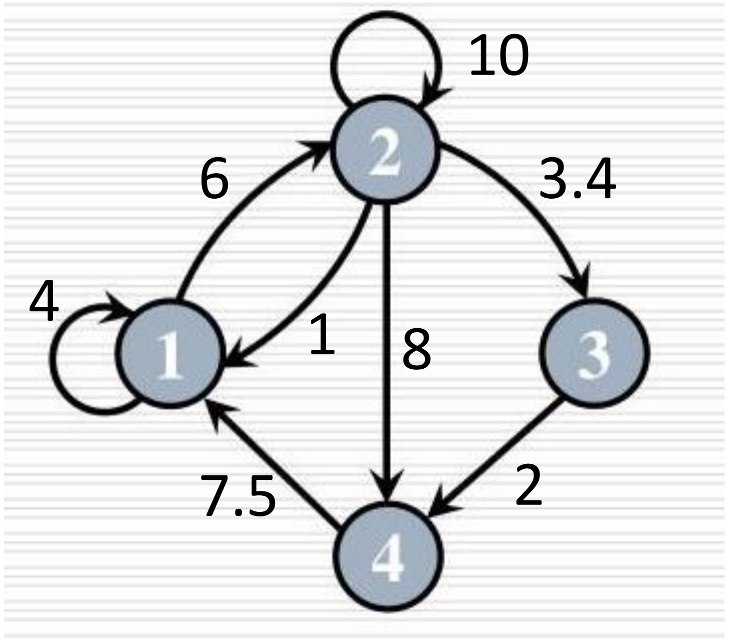




Weighted Digraphs and Matrices

There is a 1-1 correspondence between square matrices and weighted digraphs.

$$\begin{pmatrix} 4 & 6 & 0 & 0 \\ 1 & 10 & 3.4 & 8 \\ 0 & 0 & 0 & 2 \\ 7.5 & 0 & 0 & 0 \end{pmatrix} \Longleftrightarrow$$



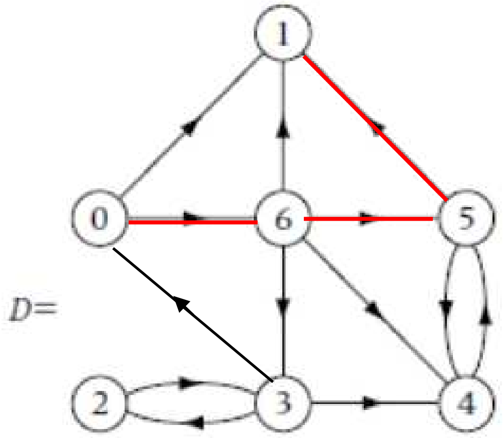
This yields and important connection between graph theory and linear algebra.

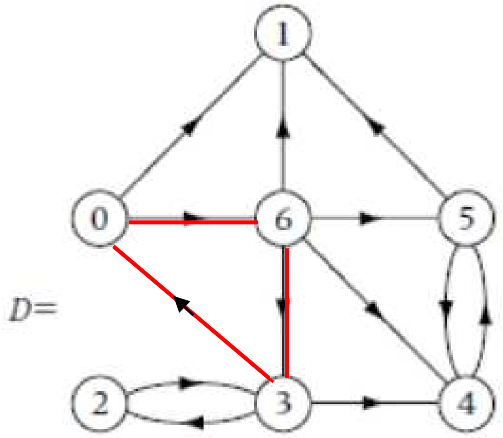
PSN. Which is more general, a graph or digraph?

Directed paths, trails, walks

Directed paths, trails, walks, cycles, circuits are defined the same for digraphs as for graphs, except that the edges must be consistently directed from the initial to the terminal vertex.

Example of a directed path from 0 to 1 of length 3 and a directed cycle of length 3:





Powers of the Adjacency Matrix and Counting Walks

- Let $D = (V, E)$ be a digraph with $V = \{1, 2, \ldots n\}$ having adjacency matrix $A = (a_{ij})_{n\times n}$.
- Let $A^k = (a_{ij}^{(k)})_{n\times n}$ denote the k^{th} power of A.

Theorem. $a_{ij}^{(k)}$ equals the number of walks from i to j of length k.

Proof of Theorem by Mathematical Induction

Basis Step. $a_{ij}^{(1)} = a_{ij}$ and there is a directed walk of length 1 from i to j iff there is an edge (i, j), so that the Theorem is true for k = 1.

Induction Step. Assume true for k, i.e., $a_{ij}^{(k)}$ is the number of directed walks of length k from i to j.

Consider the case $k + 1$. Clearly, $A^{k+1} = AA^k$.

It follows from the definition of matrix multiplication that

$$a_{ij}^{(k+1)} = a_{i1}a_{1j}^{(k)} + a_{i2}a_{2j}^{(k)} + \cdots + a_{in}a_{nj}^{(k)}$$

By the Induction Hypothesis $a_{1j}^{(k)}$ is the number of directed walks from 1 to j of length k . Since a_{i1} is 1 if there is an edge from i to j and 0 otherwise, it follows that $a_{i1}a_{1j}^{(k)}$ is the number of directed walks of length $k + 1$ from i to j that go through vertex 1. Similarly, $a_{i2}a_{2j}^{(k)}$ is the number of directed walks of length $k + 1$ from i to j that go through vertex 2, and so forth all the way up to n . It follows that $a_{ij}^{(k+1)} = a_{i1}a_{1j}^{(k)} + a_{i2}a_{2j}^{(k)} + \cdots + a_{in}a_{nj}^{(k)}$ is the total number of directed walks of length $k + 1$ from i to j. Q.E.D.

Extracurricular

BEYOND ·THE· CLASSROOM!

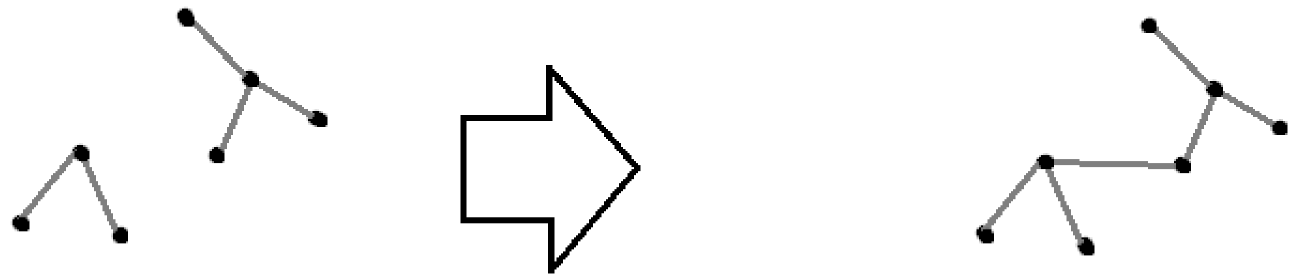
Gephi is an award-winning open-source platform for visualization and exploration for graphs, which you can download free:

https://gephi.org/

Another nice open-source graph software package is Network X:

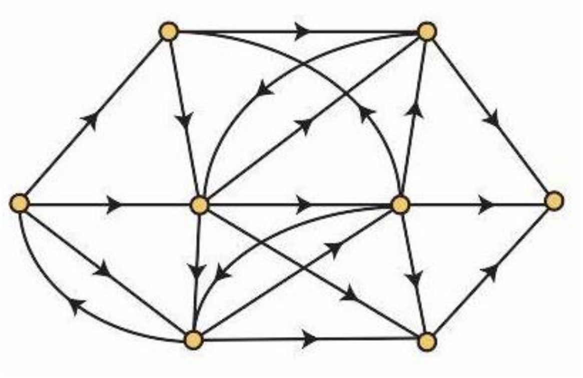
https://networkx.github.io/

Sparse Graph Joke



Deforestation:
When adding a branch gives you fewer trees.

Digraphs

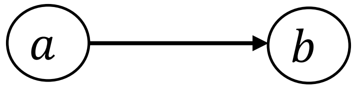


Textbook Reading:

Section 6.14 pp. 392-394
Section 6.17 pp. 399-401

Digraphs

- A digraph $D = (V, E)$ consists of a vertex set V and an (directed) edge set E, where the (directed) edges are (ordered) pairs of vertices.
- Equivalently, E is a relation on the set V, called an **adjacency relation**.
- For $(a, b) \in E$ we call a the **tail** and b the **head**.
- For convenience (a, b) can be written simply as ab.

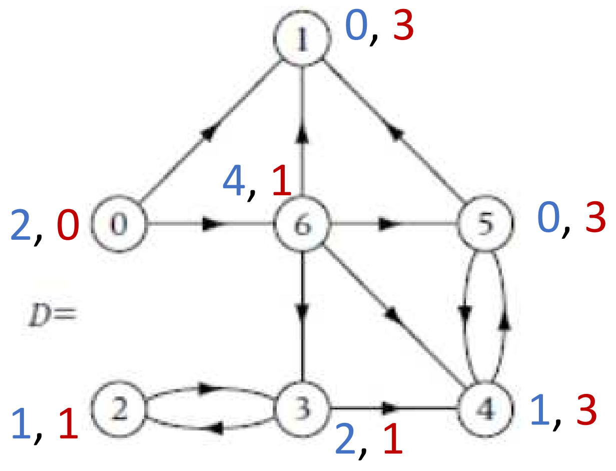


- The **out-neighborhood** of a is all vertices b that are **out-adjacent** or simply **adjacent** to a, i.e., for which there is an edge (a, b).
- The **in-neighborhood** of b is all vertices a that are **in-adjacent** to b, i.e., for which there is an edge (a, b).

Outdegree and Indegree of a vertex

The **outdegree** of a vertex v, denoted $outdeg(v)$, is the number of edges that have tail v, i.e., the cardinality of the out-neighborhood of v.

The **indegree** of a vertex v, denoted $indeg(v)$, is the number of edges that have head v, i.e., the cardinality of the in-neighborhood of v.



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

$$E = \{(0,1),(0,6),(2,3),(3,2),(3,4),(4,5),(5,1),(5,4),(6,1),(6,3),(6,4),(6,5)\}$$

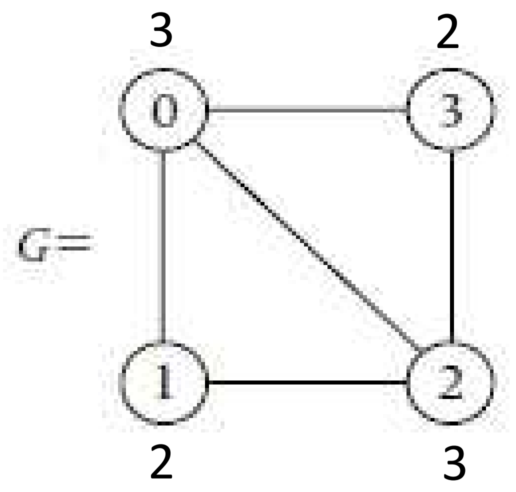
Proposition.

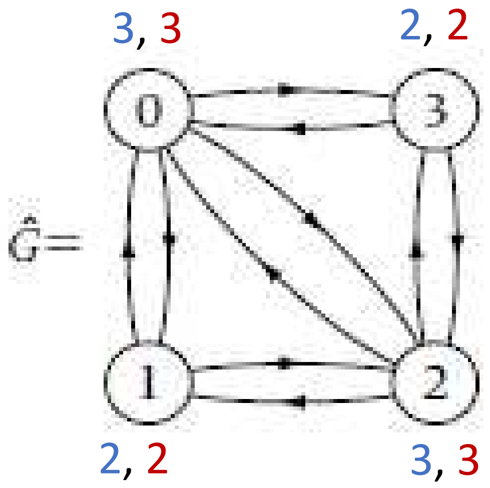
$$\sum_{v \in V} outdeg(v) = \sum_{v \in V} indeg(v) = m.$$

Proof of Proposition

- The sum of the out-degrees over all the vertices counts every edge once, i.e., each edge is counted only in the out-degree of the tail of the edge.
- Similarly, the sum of the in-degrees over all the vertices counts every edge once, i.e., each edge is counted only in the in-degree of the tail of the edge.

Note that we obtain Euler's degree formula for vertices of an (undirected) graph as a special case via that transformation to its equivalent symmetric digraph.





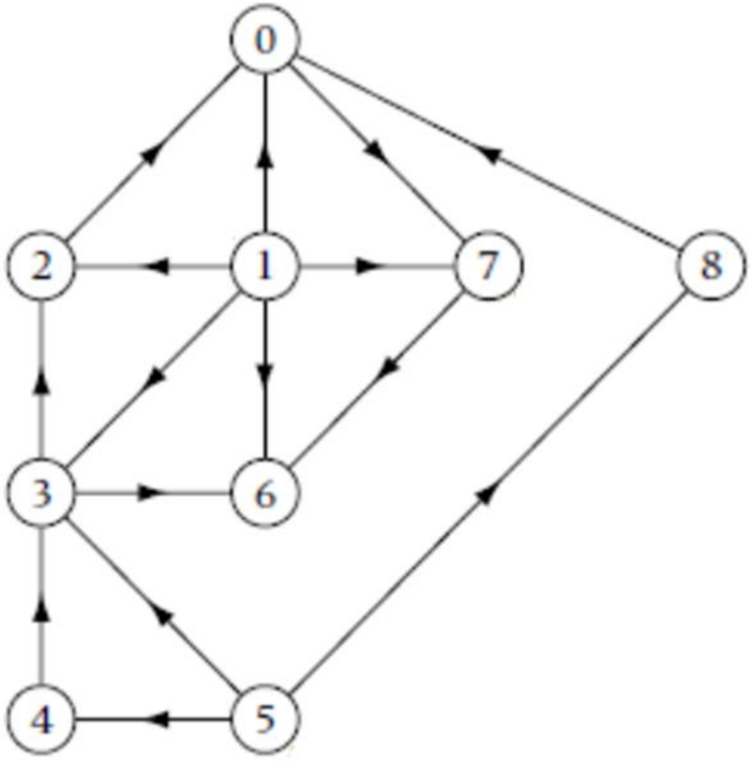
3 + 2 + 3 + 2 = 10

3 + 2 + 3 + 2 = 10

3 + 2 + 3 + 2 = 10

DAG

A **Directed Acyclic Graph** or **DAG** is a digraph without any directed cycles.



Sources and Sinks

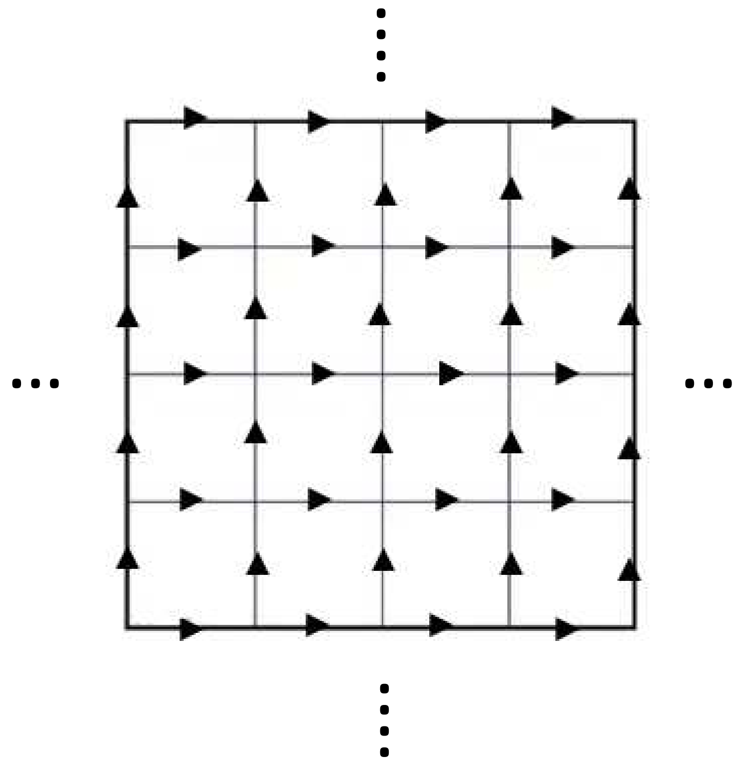
Source – a vertex *v* where all edges incident with v are directed out of *v*, i.e., in-degree(*v*) = 0

Sink – a vertex *v* where all edges incident with *v* are directed into *v*, i.e., out-degree(*v*) = 0

PSN. Show that a (finite) DAG must always contain a source and a sink.

Result not true for infinite graphs

Infinite grid with the edges oriented to the right and up is acyclic but has no source nor sink.



Round-Robin Tournament

- A round-round tournament is a tournament where every player plays every other player.
- The win-loss results (we assume no ties) can be modeled using a digraph.
- The vertices of the digraph correspond to the players.
- An edge is directed from *u* to *v* whenever player *u* defeats player *v*.
- Since tournament is round-robin, the underlying undirected graph is complete, i.e., for each pair of distinct vertices *u* and *v*, either (*u*,*v*) or (*v*,*u*) is an edge.

Example Modeling Round-Robin Tournament

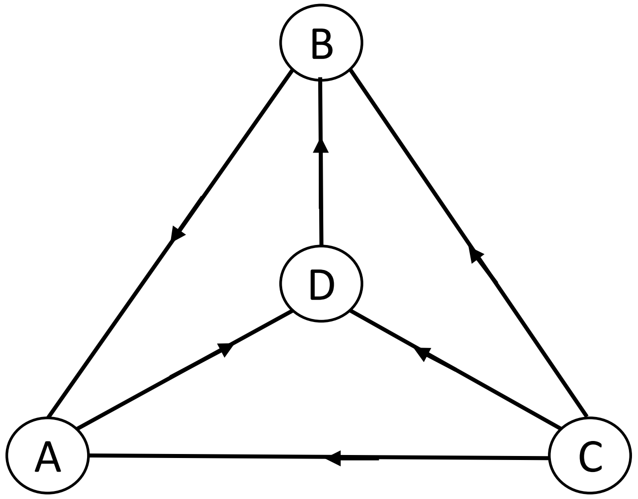
Results of round-robin tournament an digraph modeling these results:

A beats D

B beats A

C beats A, B, D

D beats B



Directed Hamiltonian Cycle

Proposition. A DAG modelling a round-robin tournament contains a directed Hamiltonian path joining the unique source (player who won all games) to the unique sink (player who lost all games).

Proof by Mathematical Induction

We perform induction on the number of vertices n.

Basis Step. A tournament of $n = 2$ vertices (players) a and b contains the directed Hamilton path ab consisting of the single edge.

Induction Step. Assume true for $n = k$, i.e., any DAG modeling a round-robin tournament with k players contains a directed Hamiltonian path from the source vertex to the sink vertex.

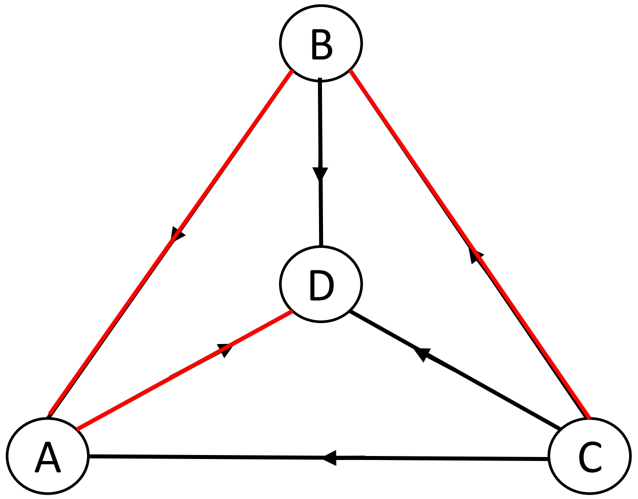
Now consider a DAG D modeling a round-robin tournament with $k + 1$ and let a and b denote the source and sink vertices, respectively.

Let D' be the DAG obtained from D by deleting vertex a and all incident edges and let a' denote the source vertex of D'. Since D' models a round-robin tournament with k players, by the **Induction Hypothesis** it contains a directed Hamiltonian path H' from a' to b.

Construct the directed Hamiltonian path from a to b in D by taking the edge aa' followed by H'. Q.E.D.

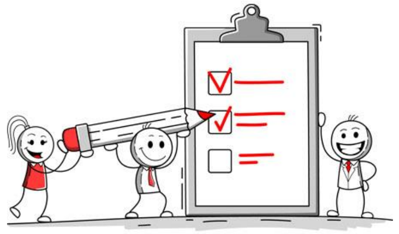
Ranking of Players

The directed Hamiltonian path determines a ranking of the players from best to worst.



Ranking: C B A D

Ordering Tasks



- *n* tasks to be performed
- Certain tasks must be performed before others.
- For example, if we are building a house, the task of pouring the foundation must precede the task of laying down the first floor. However, another pair of tasks might not need to be done in a particular order, such as painting the kitchen and painting the bathroom.
- The problem is to obtain a linear ordering of the tasks in such a way that if task *u* must be done before task *v*, then *u* occurs before *v* in the linear ordering.

Modeling with a DAG

Construct a digraph $D = (V, E)$ where V is the set of tasks and $(u, v) \in E$ whenever task *u* must precede, i.e., be performed before, task *v*.

Proof by contradiction that *D* is a DAG

Suppose D is not a DAG. Then it contains a directed cycle $v_1, v_2, \cdots, v_k, v_1$. It follows from the definition of *D* that

Task v_1 must be performed before task v_2.
Task v_2 must be performed before task v_3.

$$\vdots$$

Task v_{n-2} must be performed before task v_{n-1}.
Task v_{n-1} must be performed before task v_1.

But this implies that task v_1 must be performed before itself, a contradiction.

Modelling with a DAG cont’d

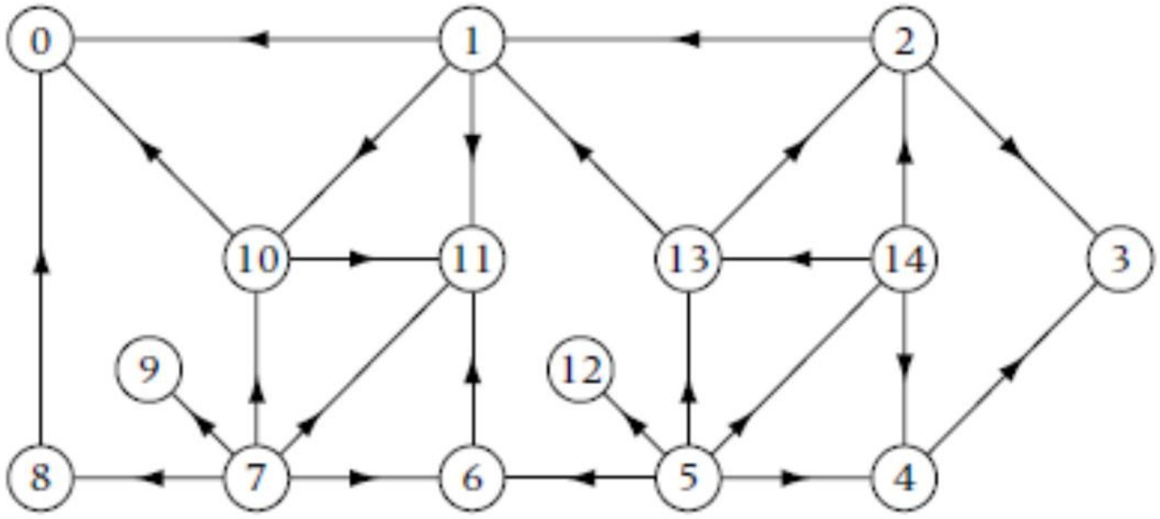
- The vertices of the DAG *D* correspond to the tasks, and a directed edge from *u* to *v* is in *D* iff task *u* must precede task *v*.
- A **topological sorting** *of D* is a listing of the vertices such that if *uv* is an edge of *D*, then *u* precedes *v* in the list.
- A **topological-sort labeling** of *D* is a labeling of the vertices in *D* with the labels 0, . . . , *n* – 1 such that for any edge *uv* in *D*, the label of *u* is smaller than the label of *v*.

Topological Sort – Straightforward Algorithm

Repeat until all vertices of DAG have been visited

1. Find a vertex *v* such that all vertices in the in-neighborhood of *v* have been visited
2. Insert *v* at the end of the list
3. Mark *v* as visited

PSN. Use straightforward algorithm to find topological sort for the following DAG:

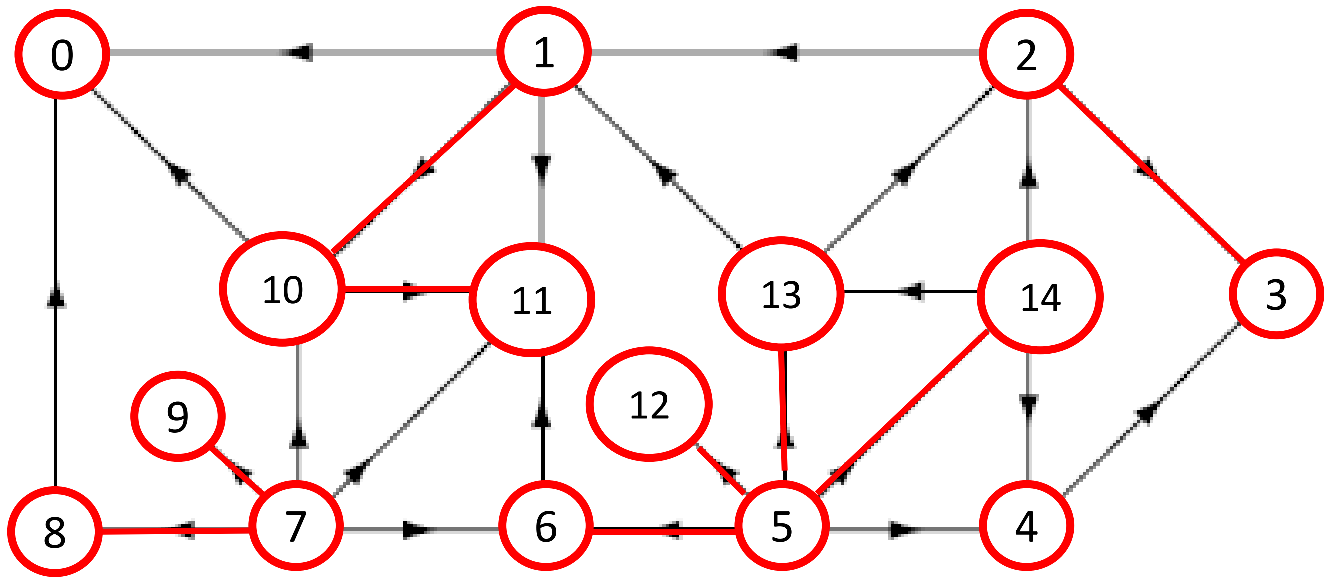


More efficient topological sort using DFT

Perform a DFT traversal keeping track of the **order in which the vertices are explored**. A vertex becomes **explored** when all the vertices in its out-neighborhood have been visited, i.e., when we backtrack from the vertex.

The topological sort order is the **reverse** of the explored order.

Action for a sample list



Topologically sorted list:

7 9 8 5 14 13 12 6 4 2 3 1 10 11 0

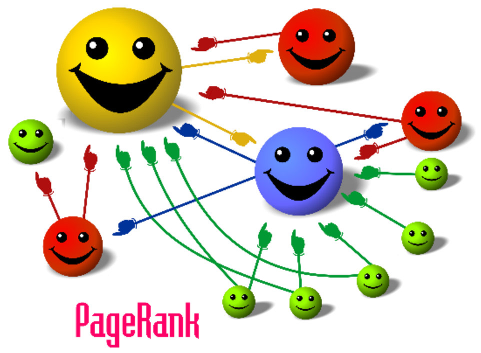
What's the best way to watch a Fly Fishing tournament ?

Live stream

PageRank

Reading:

Supplementary Notes on PageRank



Importance of Ranking Web Pages

Ranking of web pages is very important for users of the Web as well as business.

- For a given user query, there could be thousands, millions and even billions of web pages that satisfy the query.
- The ranking of the web pages effects the order in which their hyperlinks are listed.
- Most people only look at the first few pages of results for their query, sometimes only looking at the hyperlinks listed at the top of the first page.
- From the user's point of view, it is important that web pages of "high quality", "high prestige" get listed first.
- For a business, it makes a great deal of difference whether a hyperlink to their web page shows up near the beginning of the list, for example, near the top of the first page, potentially bringing in lots of new business.

PageRank

PageRank, was developed in 1996 at Stanford by Larry Page and Sergey Brin, the cofounders of Google.

Ph.D. Students at Stanford

Page and Brin were both Ph.D. students at Stanford, when they came up with the killer application of PageRank and pioneered the Google Search Engine.

Idea behind PageRank

PageRank assigns a measure of "prestige" or ranking (*PageRank*) to each web page, which is independent of any query. It is defined using a digraph **based on the hyperlink structure of the web** called the *web digraph*.

Business Flop?

They tried to sell their search engine idea using PageRank to YAHOO!®

But it was rejected

Founding of Google

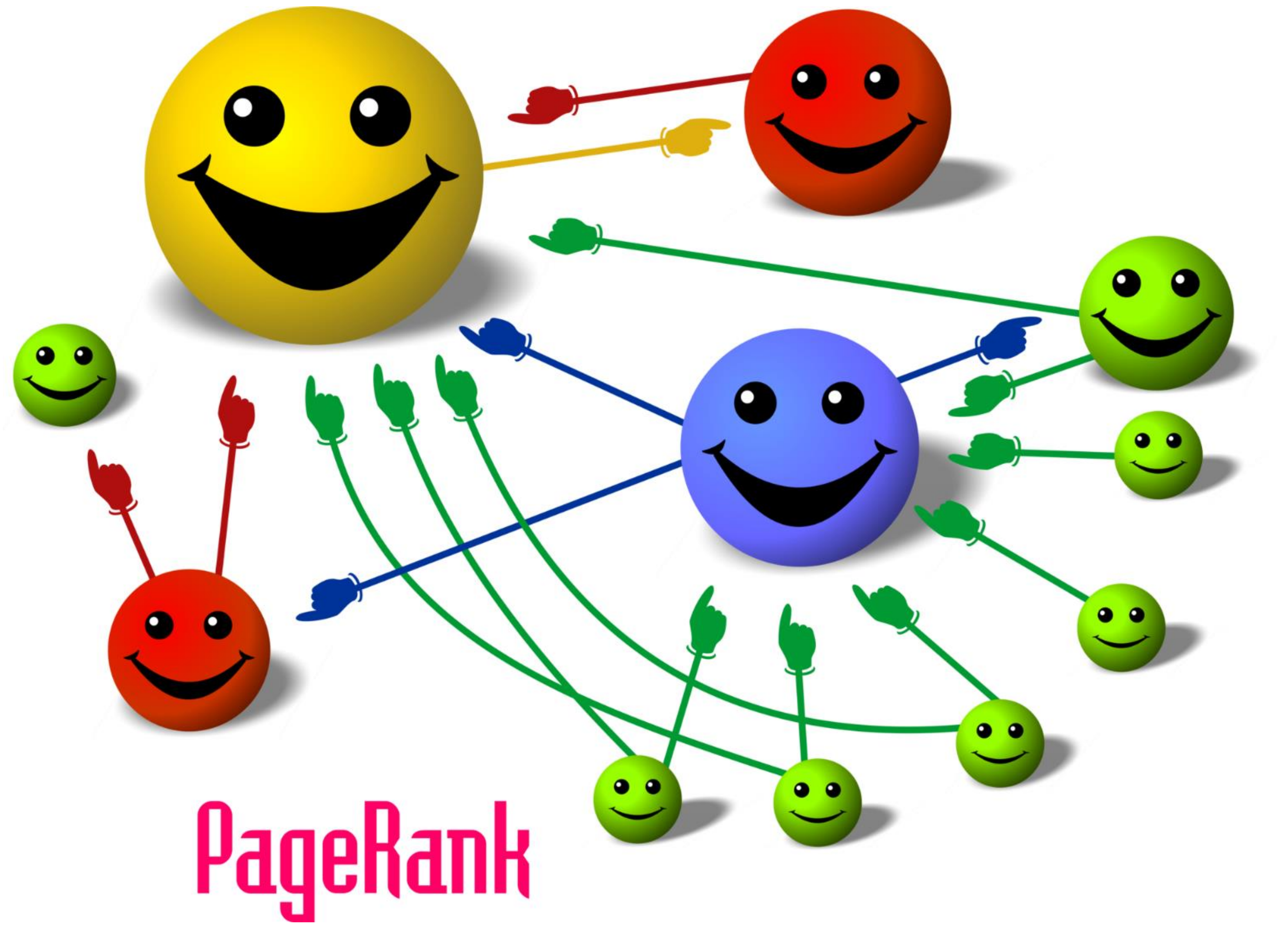
After the rejection by Yahoo, Page and Brin started their own company, which they called Google.

One of the first investors in their company was a professor at Stanford, David Cheriton, who had received his M.S. and Ph.D. degrees in computer science from the University of Waterloo. Cheriton later donated $25 million to support graduate studies and research in the School of Computer Science, subsequently renamed David R. Cheriton School of Computer Science, at the University of Waterloo.

Web Digraph

- The **web digraph** *W*
- Vertex set *V*(*W*) consists of **web pages**
- Edge set *E*(*W*) corresponds to **hyperlinks**, that is, an edge is included from page *p* to page *q* whenever there is a hyperlink reference (href) in page *p* to page *q*.

In its simplified form PageRank of a web page is measured by its in-degree in *W*



Drawback with Simplistic Definition

Using just the in-degree of a web page *p* as its rank has two weakness:

- Web pages *q* that contain a hyperlink to *p* may have different measures of prestige.
- Web pages *q* that contain a hyperlink to *p* may have different out-degrees.

If *q* has lower prestige, we don't want to count it as heavily.

Similarly, we don't want to count it as much if it has high out-degree, i.e., it includes a lot of hyperlink references.

Formula for PageRank

The PageRank $R[p]$ of web page p satisfies:

$$R[p] = \sum_{q \in N_{in}(p)} \frac{R[q]}{d_{out}(q)} \; .$$

where $d_{out}(q)$ is the **out-degree** of page *q*, or equivalently the **number *h*(*q*) of hyperlink references** that *q* contains.

PageRank has Myriad Applications

PageRank has been used for many other applications besides Google including applications to

- Bibliometrics
- Social and information network analysis
- Link prediction and recommendation.
- Systems analysis of road networks
- Gene Searching in Biology: an app called ToppGene Suite was developed at UC and Cincinnati Children's Hospital by Anil Jegga https://toppgene.cchmc.org/
- Index called pagerank-index (Pi) for quantifying the scientific impact of researchers

PageRank Concept Existed Before Google

The concept behind PageRank was not invented by Page and Brin, but was first applied by them to the hyperlink structure of the Web in the design Google. In fact, in their famous paper

The Anatomy of a Large-Scale Hypertextual Web Search Engine

they only devote one paragraph to PageRank. To quote their paper:

“Academic citation literature has been applied to the web, largely by counting citations or backlinks to a given page. This gives some approximation of a page's importance or quality.”

Computing PageRank for Mini Web Digraph *W* = (*E*,*V*)

=

$$\begin{tikzcd}
1 \arrow[r] \arrow[d, shift left] \arrow[dr, shift left] & 2 \arrow[d] \\
4 \arrow[u, shift left] \arrow[ur] \arrow[r, shift left] & 3 \arrow[l, shift left] \arrow[ul, shift left]
\end{tikzcd}$$

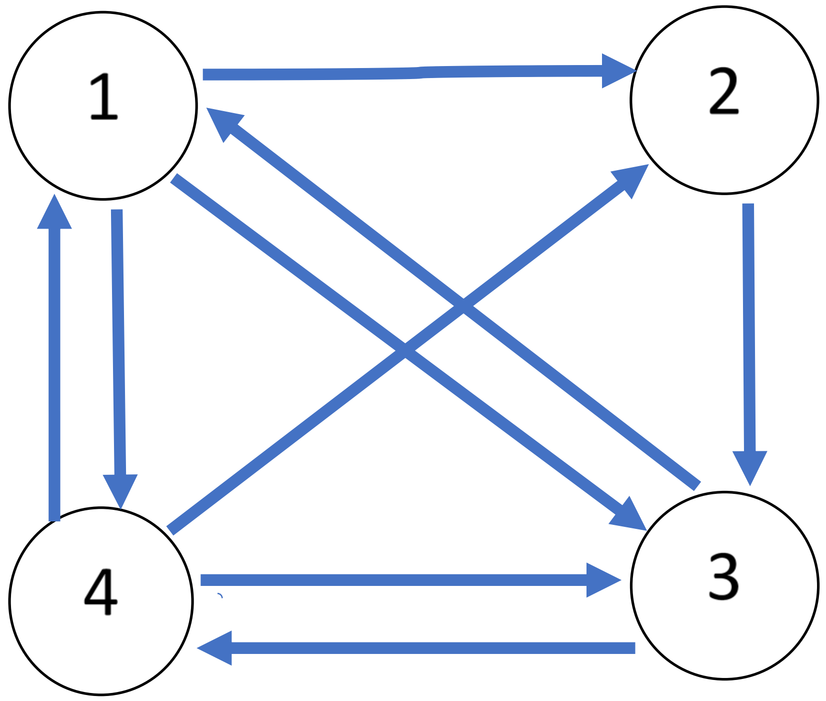
Page Rank Equations:

$$R_1 = \tfrac{1}{2}R_3 + \tfrac{1}{3}R_4$$

$$R_2 = \tfrac{1}{3}R_1 + \tfrac{1}{3}R_4$$

$$R_3 = \tfrac{1}{3}R_1 + R_2 + \tfrac{1}{3}R_4$$

$$R_4 = \tfrac{1}{3}R_1 + \tfrac{1}{2}R_3$$

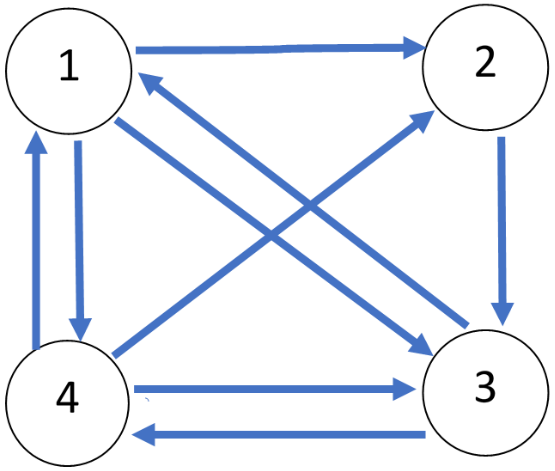


Expressing Linear Equations in Matrix Form

$$R = \begin{pmatrix} 0 & 0 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{3} & 0 & 0 & \frac{1}{3} \\ \frac{1}{3} & 1 & 0 & \frac{1}{3} \\ \frac{1}{3} & 0 & \frac{1}{2} & 0 \end{pmatrix} R, \qquad \text{where } R = \begin{bmatrix} R_1 \\ R_2 \\ R_3 \\ R_4 \end{bmatrix}$$

or equivalently,

$$R = \begin{pmatrix} 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 1 & 0 \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 \end{pmatrix}^T R$$



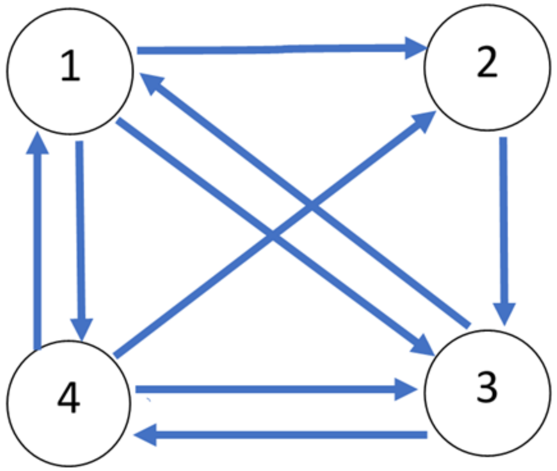
Random Walk Matrix

Let B be the matrix for a random walk on W, i.e.,

$$B[p,q] = \begin{cases} \dfrac{1}{d_{out}(p)}, & pq \in E(W), \\ 0, & \text{otherwise.} \end{cases}$$

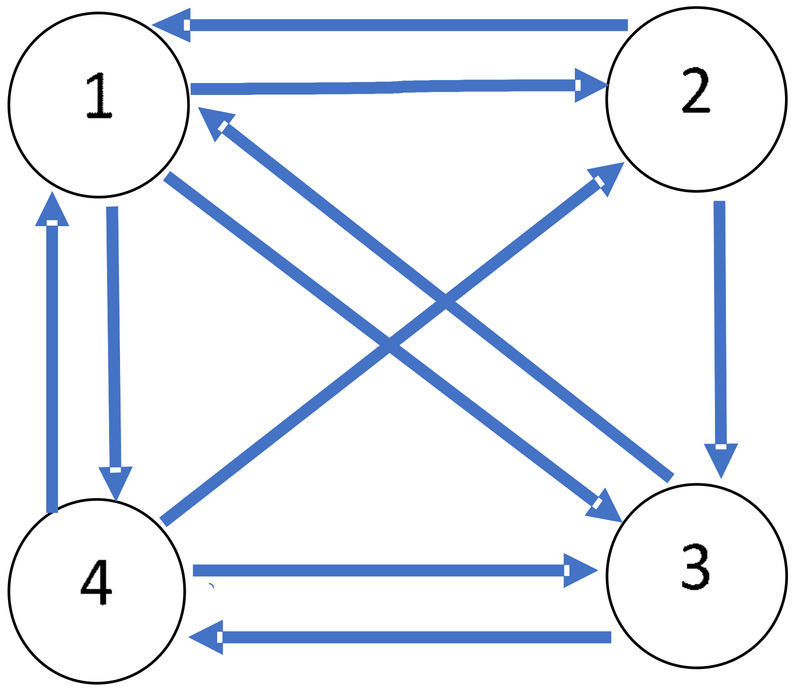
For Web, B is the matrix for a **random walk**!

$$\begin{pmatrix} 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 1 & 0 \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 \end{pmatrix}$$



PSN. For the mini-web below

a) Give the Page Rank equations
b) Give the associated matrix equation
c) Give transpose matrix and verify it is the matrix for a random walk.



PageRank is Principal Eigenvector of Matrix of a Random Walk on *W*

Letting be the PageRank *R* be the vector or array, i.e., *R*[*p*] is the PageRank of Page p, the formula for PageRank is

$$R = B^{\mathsf{T}} R.$$

R is an eigenvector of the matrix *B* (and B^{T}) for the eigenvalue 1. Since 1 is the largest eigenvalue of B, *R* is **principal eigenvector**.

Standard Linear Algebra Algorithms to Compute Principal Eigenvector

Using standard linear algebra algorithms to compute Principal Eigenvector is **prohibitive** for large matrices.

At the present time, the last realistic estimate of the World Wide Web was about **19.2 billion** web pages. It is not possible to store a matrix of this size, because the number of entries is **astronomical**!

368,640,000,000,000,000,000

Efficient Algorithm by Iterating

$$R_i = B^{\top} R_{i\text{-}1},\ i = 1, 2, \ldots$$

or equivalently,

$$R_i[p] = \sum_{q \in N_{in}(p)} \frac{R_{i-1}[q]}{d_{out}(q)}\ , i = 1, 2, \ldots$$

where R_0 can chosen to be a vector of nonnegative reals whose entries sum to 1.

Random Walk Interpretation of PageRank

By a simple induction argument, we obtain

$$R_i = (B^\top)^i R_0,\ i = 1, 2, \ldots$$

By a theorem of Markov, $B^i[p,q]$ equals the probability of a random walk on W starting a page p and ending up at page q after i steps.

Thus,

$(B^\top)^i[p,q] = B^i[q,p]$ = probability that a random walk starting at page q will end up at page p after i steps.

Equivalent Interpretation

$B^i[p][q]$ is the probability that an aimless web surfer starting page p reaches q in i steps (by following a path of i hyperlinks).

Equivalently,

$(B^{\mathrm{T}})^i[p][q]$ is the probability that the aimless web surfer starting a page q will reach p after i steps.

Random (Aimless) Web Surfer and PageRank

Since the entries of R_0 are positive and sum to 1, R_0 determines a probability distribution on the set of pages $V(W)$, where $R_0[q]$ is the probability that the aimless surfer begins surfing from page q.

Then $R_i[p] = (B^T)^i R_0[p]$ is the probability the surfer will end up on page p after i steps.

Interesting Special Case

For a particular page q, e.g., your web page, set $R_0[q] = 1$ and set $R_0[p] = 0$, for all $p \neq q$.

Then $R_i[p] = (B^\top)^i R_0[p]$ is the probability that a random surfer starting at page q will end up at page p after i steps.

Conditions for convergence

The vector R_i will not necessarily converge to the principle eigenvector R unless the digraph W satisfies certain conditions.

Condition 1. ***W*** is **strongly-connected.** There is a directed path from p to q for every two vertices p and q.

Condition 2. ***W*** **is aperiodic**. There is some integer N, such that for all $k \geq N$, W contains a closed walk of length k starting at any given vertex.

Damping Factor

The actual World Wide Web Digraph is neither strongly-connected nor aperiodic. To ensure that the iteration for PageRank converges it is necessary to introduce a **damping factor**.

Let *n* denote the number of nodes of the web digraph W. The PageRank *R*[*p*] of a web page *p* is given by:

$$R[p] = \frac{1 - d}{n} + d \sum_{q \in N_{in}(p)} \frac{R[q]}{d_{out}(q)} \ ,$$

where *d* is the damping factor between 0 and 1.

Iteration Converges with Damping Factor

The damping factor is equivalent to adding edges for every pair of web pages (*x*,*y*) giving it a the very small probability $\frac{1-d}{n}$ and slightly reducing the probability of each original edge having tail *q* from the $\frac{1}{d_{out}(q)}$ to $\frac{d}{d_{out}(q)}$.

The underlying digraph is now complete, so that it is necessarily strongly connected and aperiodic.

This means the iteration for PageRank will always converge when a damping factor is added!

Computing PageRank in Practice for the World Wide Web

In practice compute PageRank by iterating following formula

$$R_i[p] = \frac{1-d}{n} + d \sum_{q \in N_{in}(p)} \frac{R_{i-1}[q]}{d_{out}(q)} \ , i = 1, 2, \ldots$$

Empirical experiments have shown that acceptable ranking functions R_i are achieved in 52 iterations for about 322 million hyperlinks.

Taking the damping factor *d* to be between .8 and .9 has been found to work well in practice.

Random Web Surfer with Damping Factor

As before the value of PageRank after *i* iterations $R_i[p] = (B^T)^i R_0[p]$ is the probability that a random surfer will end up on page *p* after *i* steps.

The difference is that the surfer can **randomly jump from a page *q* to an arbitrary page, but with a very small probability, i.e.,** $\frac{1-d}{n}$. Otherwise, the surfer **randomly goes with equal probability (** $\frac{d}{d_{out}(q)}$ **) to a page in the out-neighborhood of *q, i.e.,* clicks on a hyperlink on page *q*.**

What do frogs say that surf the internet?

Reddit reddit.

Intro to Counting and Combinatorics

Textbook Reading

Sections 7.1-7.5, pp. 421-442

Traveling Salesperson Problem

- A salesperson's territory includes n cities that must be visited on a regular basis.
- Between each pair of cities, air service is available.
- The problem is to schedule a sequence of flights that visits each city exactly once before returning to the starting point.
- Now suppose there is a weight associated with each city representing the time
- Find schedule so that the total time spent flying is minimized.

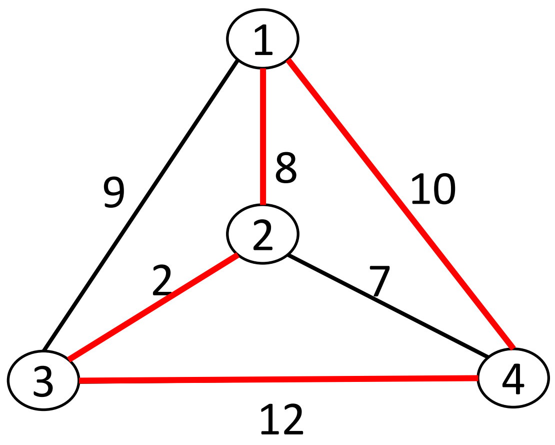
Solution

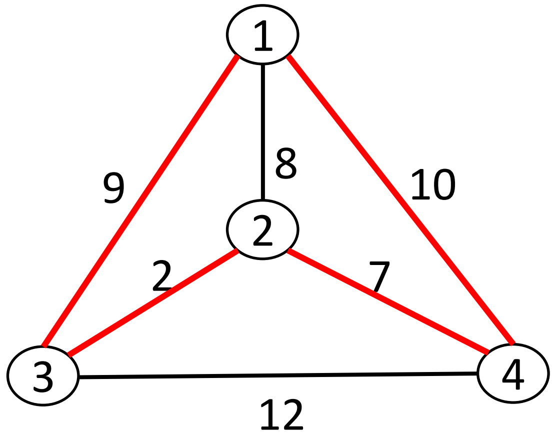
Equivalently, the problem is to find a **Hamiltonian cycle** of minimum weight.

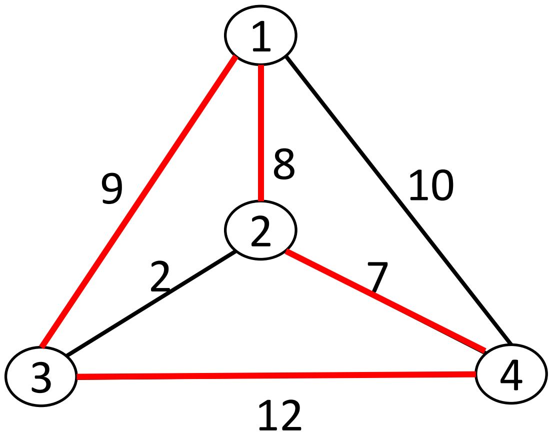
We can enumerate all such cycles as follows. Starting with city 1, there are $n-1$ ways to choose the second city to visit, then, $n-2$ to choose the third city, $n-3$ to choose the fourth city, and so forth.

It follows that the number of Hamiltonian cycles starting and ending with city 1 is

$$(n-1)\times(n-2)\times(n-3)\times\cdots\times 2\times 1=(n\ -1)!$$



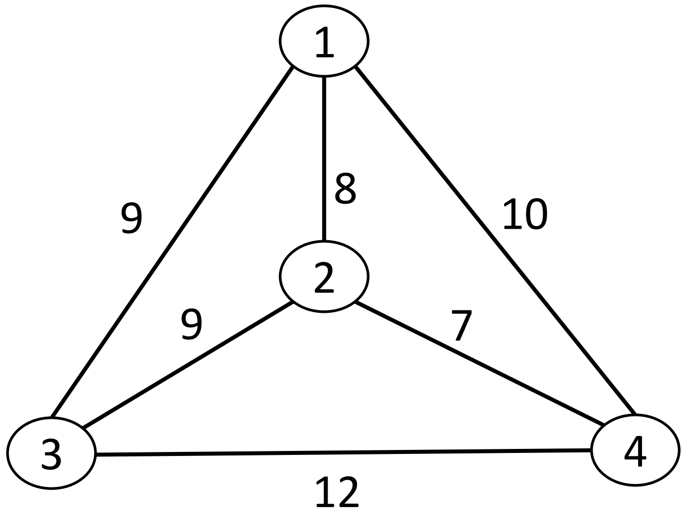




12341 8+2+12+10 = 32	**13241 9+2+7+10 = 28**	12431 8+7+12+9 = 36
14321 10+12+2+8 = 32	**14321 10+7+2+9 = 28**	13421 9+12+7+8 = 36

There are 6 Hamiltonian cycles with initial vertex 1.
There are 3 when paired with cycle traversed in reverse.
Minimum weight Hamiltonian cycle has weight 28.

PSN. Solve the TSP for sample weighted graph



Counting Hamiltonian Cycles

Brute force algorithm is to enumerate all Hamiltonian cycles to find the one of smallest weight. Starting with vertex 1, there are

n – 1 choices for second vertex

n – 2 choices for third vertex

⋮

1 choice for *n*th vertex

Total number of choices altogether is

$$(n-1) \times (n-2) \times \cdots \times 2 \times 1 = (n-1)!$$

This is prohibitively large computationally for even relatively small *n*.

The TSP is at least as hard as the problem of finding a Hamiltonian cycle, so it is NP-hard.

Combinatorial Explosion

There is a combinatorial explosion in enumerating the number of Hamiltonian cycles.

The Traveling Salesperson Problem (TSP) includes the Hamiltonian cycle problem as the special case where we assign each edge a weight of 1 if it is in the graph and M, where M is very large, i.e., M > n, otherwise. Thus, TSP is NP-hard.

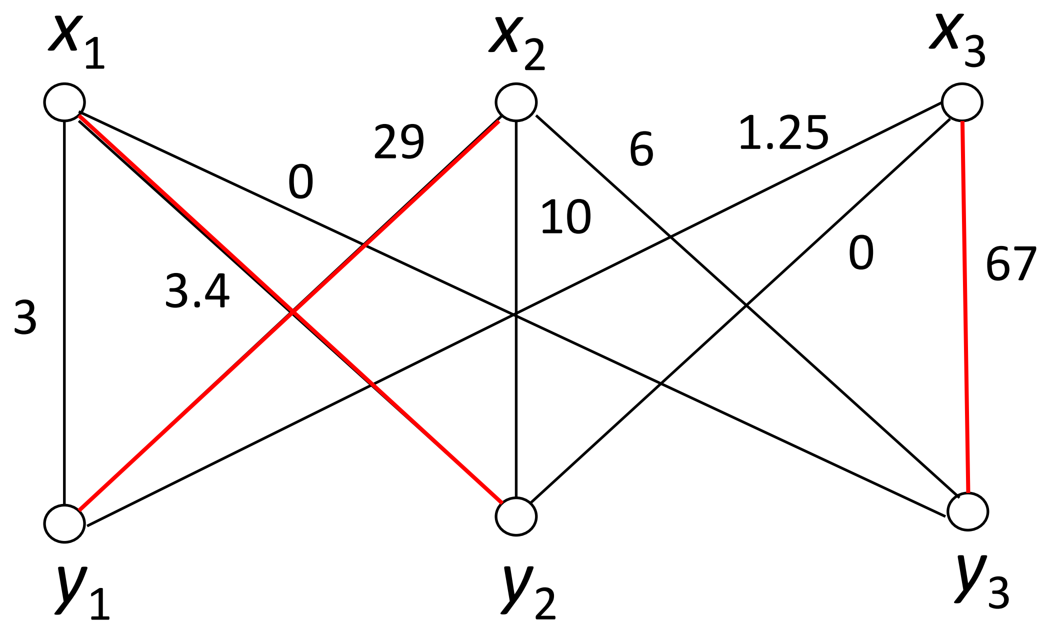
Perfect Matchings in Weighted Bipartite Graphs

Weighted complete bipartite graph:

$G = (V,E)$ with vertex bipartition $V = X \cup Y$, where $X = \{x_1, \ldots, x_{n-1}\}$ and $Y = \{y_1, \ldots, y_n\}$.

Each edge $x_i y_j$ of G is assigned the a real weight ω_{ij}, $i, j \in \{1, \ldots, n\}$.

Sample weighted complete bipartite graph



It can be input using the two-dimensional array (matrix):

$$\begin{pmatrix} 3 & \color{red}{3.4} & 0 \\ \color{red}{29} & 10 & 6 \\ 1.25 & 0 & \color{red}{67} \end{pmatrix}$$

A perfect matching corresponds to a traversal of the matrix.

Matchings

A **matching** is a set of edges that have no vertex in common. A **perfect** matching is one that spans the vertices.

The **weight** of a matching *M*, denoted by $\omega(M)$, is the sum of the weights of its edges, i.e.

$$\omega(M) = \sum_{e \in M} \omega(e).$$

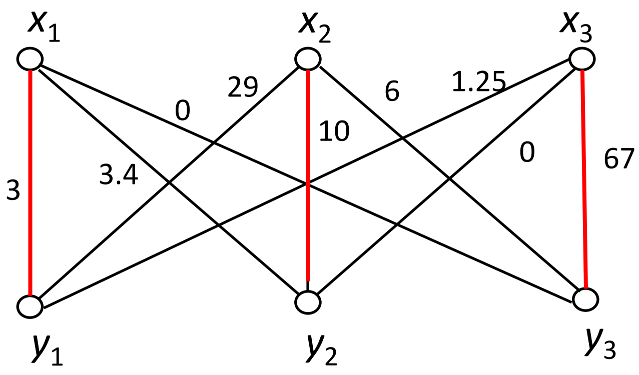
Maximum Weight Perfect Matching

A **maximum-weight perfect matching** is one that maximizes $\omega(M)$.

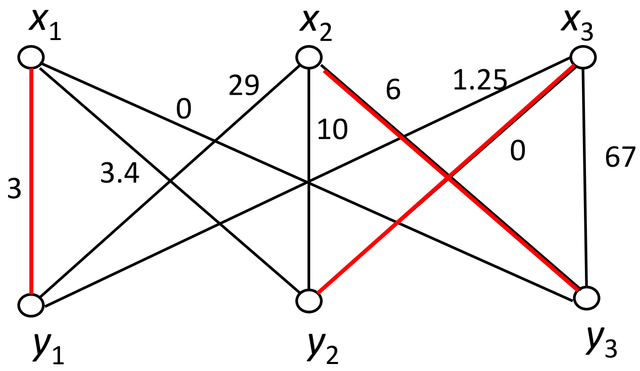
There are many natural applications for finding a maximum weight perfect matching. For example, finding an assignment of workers to jobs so that the total effectiveness of the workers is optimized.

Maximum Perfect Matching in Sample Graph

3+10+67 =80

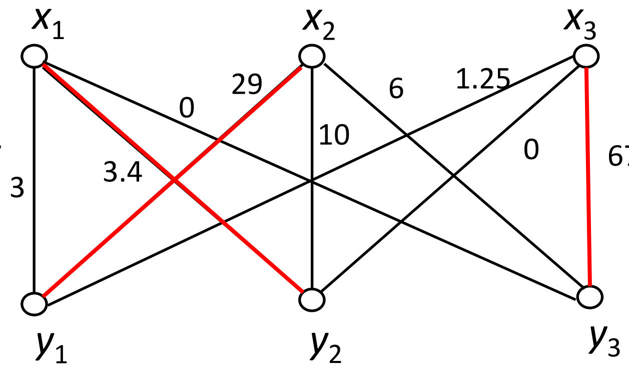


3+6+0 =9

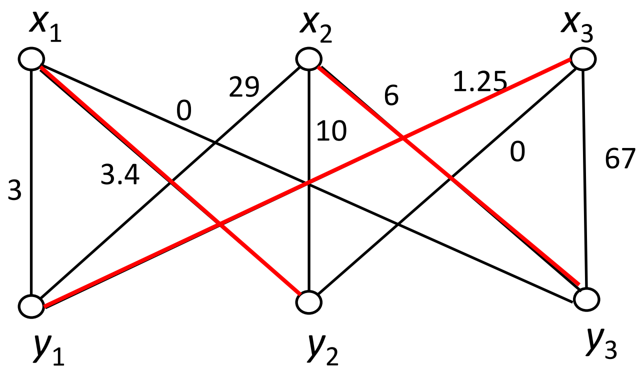


3.4+29+67 = 99.4

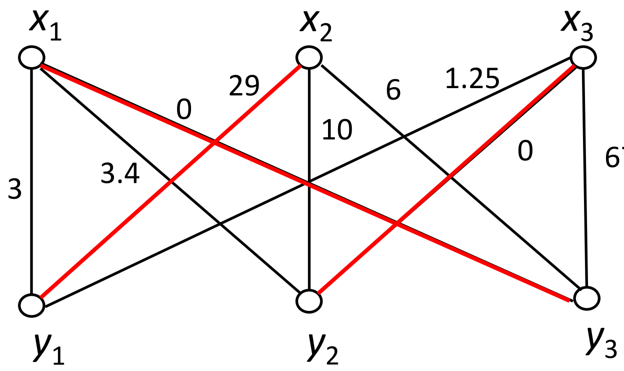
maximum



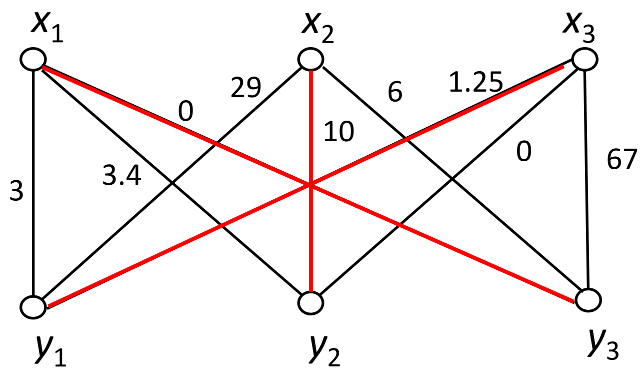
3.4+6+1.25 = 10.65



0+29+0 =29



0+10+1.25= 11.25



Brute force very inefficient

A brute-force algorithm that enumerates all $n!$ perfect matchings and chooses one of maximum weight is hopelessly inefficient.

There is a combinatorial explosion of perfect matchings.

This problem can be solved efficient using an algorithm due to Kuhn-Munkres called the Hungarian Algorithm.

Intro to Combinatorics and Counting

A permutation of a set S is a bijective mapping from S to itself.

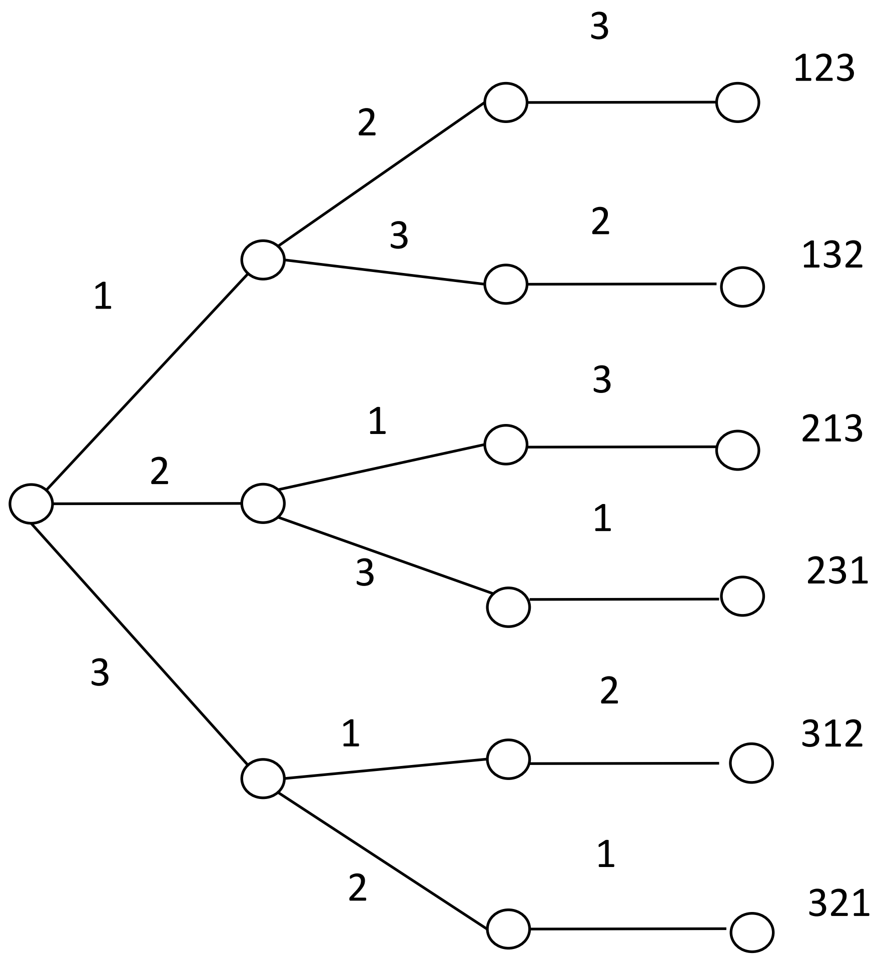
For example $S = \{1, 2, 3\}$

Permutation are:

$\begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix}$ $\begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \end{pmatrix}$ $\begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix}$ $\begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix}$ $\begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{pmatrix}$ $\begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix}$

123 132 213 231 312 321

Tree enumerating all permutations of 3 elements



Number of Permutations of n Elements

Proposition. The number of permutations of an n-element set is $n!$

Proof using tree enumerating choices:

- The number of nodes on the first level is n.
- Each of those n nodes has $n-1$ children, so there are $n(n-1)$ nodes on the second level.
- Each of these $n(n-1)$ nodes on second level has $n-2$ children, so there are $n(n-1)(n-2)$ nodes on the third level.
- This continues until we reach the n^{th} level with $n \times (n-1) \times \cdots \times 1 = n!$ (leaf) nodes, one for each permutation.

The Multiplication Principle

For a procedure of m successive distinct and independent steps with

n_1 outcomes possible for the first step,

n_2 outcomes possible for the second step,

$$\vdots$$

n_m outcomes possible for the m^{th} step,

the total number of possible outcomes is

$$n_1 \times n_2 \times \cdots \times n_m$$

In the special case of counting permutations of an n-element set

$$n_i = n - i, i = 1, 2, \dots, n.$$

The Addition Principle

For a collection of m disjoint sets with n_1 elements in the first, n_2 elements in the second **…** , and n_m elements in the m^{th}, the number of ways to choose one element from the collection is

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

Mom's Diner

Main Course

1. Hamburger
2. Chicken
3. Hotdog
4. Special

Drinks

I. Fizzy Orange
II. Jug of Tea

Desert

A. Ice Cream Cone
B. Sunday
C. Fudge

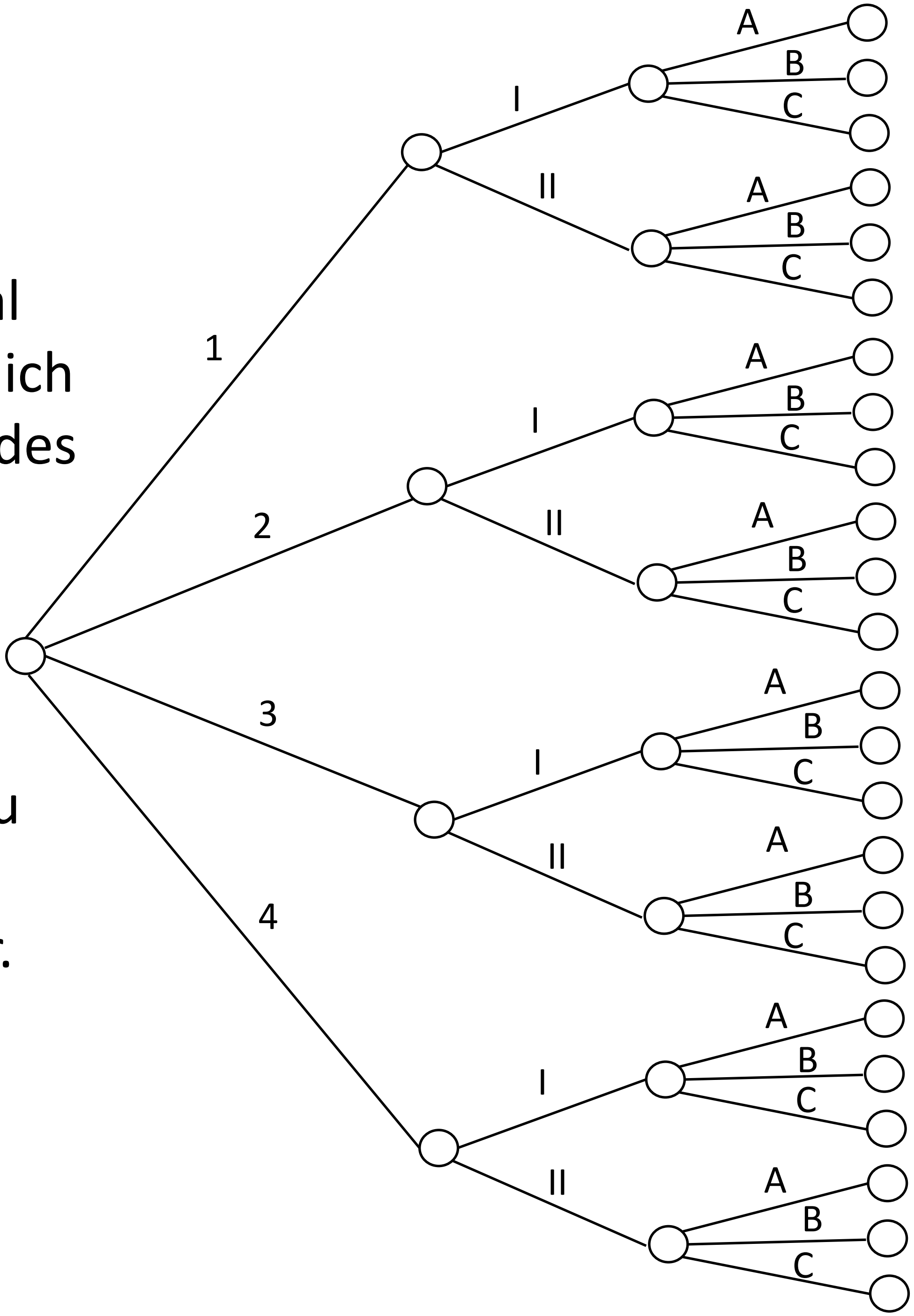
Using the Addition Principle to compute the number of ways of choosing one item from the collection, which equals total number of items available, is given by

$$4 + 2 + 3 = 9$$

Using the **Multiplication Principle** to compute the total number of combinations, which equals the number of leaf nodes of the tree:

$$4 \times 2 \times 3 = 24$$

Conclusion: There are 9 menu items and 24 different meal combinations at Mom’s Diner.



Lexicographic order

The set of all $n!$ permutations can be linearly order as follows.

Given any permutations two π_1 and π_2, let i be the first positive where they disagree.

Then π_1 is **lexicographically smaller** than π_2, written $\pi_1 < \pi_2$, iff $\pi_1(i) < \pi_2(i)$.

Examples:

546123 < 546132, 512364 < 612345, 231564 < 231645

Counting the Complement

- Sometimes it easier to compute the complement.
- For example, consider the problem of computing all permutations of the numbers from {1, 2, …, 8}, so that 1 and 2 do not occur together, i.e., in consecutive positions.
- To count the number of permutations where they occur together first place them in consecutive positions. There are 14 ways of doing this, i.e., in positions 1 and 2, positions 2 and 3, …, position 7 and 8 and there are two ways of placing them with 1 first and 2 second or vice versa.
- After 1 and 2 are placed, there are 6! ways of placing the remaining numbers, yielding a total of $14 \times 6!$.
- Thus the number of permutations where 1 and 2 are not placed in consecutive positions equals the total number of permutations minus the number where they occur together, i.e.,

$$8! - 14 \times 6! = 40320 - 14 \times 720 = 30240.$$

$\boldsymbol{r}$-permutation of an $\boldsymbol{n}$-element set

An $\boldsymbol{r}$**-permutation** of an n-element set is a linear ordering of r elements of the set.

Let $P(n,r)$ denote the number of all r-permutations of an n-element set.

Using the Multiplicative Principle we obtain the following formula for $P(n,r)$.

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

Example

{ 123, 132, 213, 231, 312, 321,
124, 142, 214, 241, 412, 421,
125, 152, 215, 251, 512, 521,
134, 143, 314, 341, 413, 431,
135, 153, 315, 351, 513, 531,
145, 154, 415, 451, 514, 541,
234, 243, 324, 342, 423, 432,
235, 253, 325, 352, 523, 532,
245, 254, 425, 452, 524, 543,
345, 354, 435, 453, 534, 543 }

There are 60 3-permutations of a 5-element set enumerated above.

Using the Multiplicative Principle to compute we obtain:

$$P(5,3) = 5 \times 4 \times 3 = 5!/2!$$

Circular Permutations

How many ways are there of seating n people at a circular table, so that in no two arrangements everyone has the same person on the left-hand and right-hand sides? Two seating arrangements are considered the same if one could be obtained from the other, by moving people in a clockwise direction around the table.

Seat the first person anywhere.

After that there are $(n-1)!$ ways to seat the remaining people.

Combinations

An unordered selection of r elements from an n element set is called a **combination**.

Let $C(n, r)$ denote the number of combinations of r elements selected from a set of n elements, i.e., the number of subsets of size r. It is also denoted by $\binom{n}{r}$.

Combinations cont'd

{ 123, 132, 213, 231, 312, 321,	{1,2,3}
124, 142, 214, 241, 412, 421,	{1,2,4}
125, 152, 215, 251, 512, 521,	{1,2,5}
134, 143, 314, 341, 413, 431,	{1,3,4}
135, 153, 315, 351, 513, 531,	{1,3,5}
145, 154, 415, 451, 514, 541,	{1,4,5}
234, 243, 324, 342, 423, 432,	{2,3,4}
235, 253, 325, 352, 523, 532,	{2,3,5}
245, 254, 425, 452, 524, 543,	{2,4,5}
345, 354, 435, 453, 534, 543 }	{3,4,5}

Note that the 3-permutations of a 5-element set corresponds to ordered subsets, where each unordered set is counted 3! = 6. It follows that

$$C(5,3) = \frac{P(5,3)}{3!} = \frac{60}{6} = 10.$$

General Formula

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

Card Hands

How many 5-card poker hands are there?

How many 13-card bridge hands are there?

The deck contains 52 cards. The number of ways of choosing a 5 cards is

$$C(52,5) = \frac{52!}{5!\,48!} = \frac{52 \times 51 \times 50 \times 49 \times 48}{5!} = 2{,}598{,}960$$

Number of 13-card bridge hands is

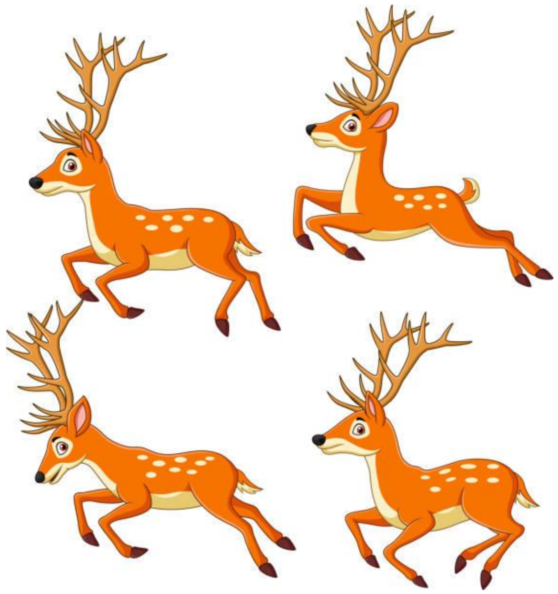
$$C(52,13) = \frac{52!}{13!\,39!} = 635{,}013{,}559{,}600$$

PSN.

a) How many permutations of 7 elements are there?
b) How many ways are there of seating 10 people at a circular table, so that in no two arrangements everyone has the same person on the left-hand and right-hand sides?
c) How many 5-permutations of an 8-element set are there?
d) How many subsets of size 5 of an 8-element set are there?

What did the footwear salesperson do to get the deer away from his house?

He shoe'd them away.



Permutations and Combinations

Textbook Reading

Chapter 7, Sections 7.5, 7.6, 7.8, pp. 436-457

Computing the kth permutation

Note that the first $\frac{n!}{n} = (n-1)!$ permutations begin with 1, the next $(n-1)!$ permutations begin with 2, and so forth. This can be used to compute the position of the k^{th} permutation in lexicographical order.

Example. Find the 52nd permutation π of 1, 2, 3, 4, 5.

- There are $5! = 120$ permutations. The first $4! = 24$ begin with a 1. The next 24 begin with a 2. It follows that π is the 4th permutation to begin with a **3**.
- Repeat this for the 4th permutation of the 4 numbers 1, 2, 4, 5. The first 3! = 6 permutations begin with a 1. Therefore, there is a **1** in the second position of π.
- Repeat for the 4th permutation of the 3 numbers 2, 4, 5. The first 2! = 2 permutations begin with a 2, the second 2 permutations begin with a 4. Therefore there is a **4** in the third position of π and it was the 2nd permutation with this property.
- Repeat for the 2nd permutation of 2 the numbers 2, 5. It follows that the last two positions of π are **5** then **2**.
- Combining all steps the 52nd permutation π is **31452**

Generating a Random Permutation

```
procedure Permute($L[0:n - 1]$)
Input:     $L[0:n - 1]$ (an array of list elements)
Output: $L[0:n - 1]$ (an array of list elements randomly
                                                permuted)
    for $i = 0$ to $n - 2$ do
        $j \leftarrow Random(i, n - 1)$
        interchange($L[i], L[j]$)
    endfor
end Permute
```

Random(*i*, *j*) returns a random number (index) between *i* and *j.*

Card Hands

How many 5-card poker hands are there?

How many 13-card bridge hands are there?

The deck contains 52 cards. The number of ways of choosing a 5 cards is

$$C(52,5) = \frac{52!}{5!\,48!} = \frac{52 \times 51 \times 50 \times 49 \times 48}{5!} = 2{,}598{,}960$$

Number of 13-card bridge hands is

$$C(52,13) = \frac{52!}{13!\,39!} = 635{,}013{,}559{,}600$$

How many poker hands are there having (exactly) one pair?

- # of choices of rank for pair is $C(13,1) = 13$
- # of choices for suits for pair is $C(4,2) = (4\times3)/2! = 6$
- # of choices of ranks for remaining 3 cards different from rank of pair is $C(12,3) = (12\times11\times10)/3! = (12\times11\times10)/6 = 220$
- # of choices of suits for remaining 3 cards is $C(4,1) \times C(4,1) \times C(4,1) = 4\times4\times4 = 64$

Number of hands having one pair is

$C(13,1) \times C(4,2) \times C(12,3) \times C(4,1) \times C(4,1) \times C(4,1)$

$= 13 \times 6 \times 220 \times 4 \times 4 \times 4 = 1098240$

How many Full Houses are there?

A Full House consists of a pair and three of a kind.

- # of choices of rank for pair is $C(13,1) = 13$
- # of choices of suits for pair is $C(4,2) = (4\times3)/2! = 6$
- # of choices of rank for three of a kind is $C(12,1) = 12$
- # of choices of suits for three of a kind is $C(4,3) = 4$

Number of Full House hands is

$C(13,1) \times C(4,2) \times C(12,1) \times C(4,3) = 13 \times 6 \times 12 \times 4 = 3744$

PSN. How many poker hands with two pairs (of different ranks) are there?

How many bridge hands are there with a void in one suit?

A bridge hand consists of 13 cards. If there is no cards of a certain suit, we way that there is a void in that suit.

# of bridge hands not containing a particular suit is $C(39,13)$.

# of suits is $C(4,1) = 4$

# of bridge hands with a void in one suit is

$C(39,13) \times C(4,1)$

PSN. How many bridge hands have a void in two suits?

Permutations with Repetitions

How many permutations of *cincinnati* are there?

c is repeated 2 times
i is repeated 3 times
n is repeated 2 times
a occurs 1 time
t occurs 1 time

- *cincinnati* has a total of 10 letters.
- So there are 10! permutations.
- However many of these permutations are identical.
- We must take into account and divide by the amount of over counting

Replacing cincinnati with $c_1i_1n_1c_2i_2n_2n_3a_1t_1i_3$

There are 10! distinct permutations
There are 2! ways of permutation the c_i's
There are 3! ways of permutation the i_i's
There are 3! ways of permutation the n_i's
There are 1! ways of permutation the a_i's
There are 1! ways of permutation the t_i's

Therefore, there are 2!3!3!1!1! ways of permutating all of them. Thus, we are overcounting by a factor of 2!3!3!1!1! when counting the 10! permutations of cincinnati

It follow that the number of permutations of cincinnati is

$$\frac{10!}{2!\,3!\,3!\,1!\,1!}$$

An alternative solution

Start with 10 slots, one for each letter of *cincinnati*. A permutation of the 10 letters of *cincinnati* can be obtained by

Choose 2 of the 10 slots for *c*.

Choose 3 of the remaining 8 slots for *i*.

Choose 3 of the remaining 5 slots for *n*.

Choose 1 of the remaining 2 slots for *a*.

Choose 1 of the remaining 1 slots for *t*.

Using the Multiplication Principle the number of ways of doing this is

$$C(10,2) \times C(8,3) \times C(5,3) \times C(2,1) \times C(1,1)$$

$$= \frac{10!}{2!8!} \times \frac{8!}{3!5!} \times \frac{5!}{3!2!} \times \frac{2!}{1!1!} \times \frac{1!}{1!0!} = \frac{10!}{2!3!3!1!1!}$$

Suppose we have k symbols distinct symbols, where the i^{th} symbol has multiplicity $r_i, i = 1, \dots, k$ and the total number of symbols with repetition is $n = r_1 + r_2 + \cdots + r_k$. Let $P(n; r_1, r_2, \dots, r_k)$ denote the number of permutations with repetitions.

Theorem. $P(n; r_1, r_2, \dots, r_k) = \dfrac{n!}{r_1!r_2!\cdots r_k!}$.

Proof

Take n slots.

Choose r_1 of the n slots for 1st symbol.

Choose r_2 of the remaining $n - r_1$ slots for 2nd symbol.

Choose r_3 of the remaining $n - r_1 - r_2$ slots for 3rd symbol.

$$\vdots$$

Choose r_k of the remaining $n - r_1 - \cdots - r_{k-1}$ slots for kth symbol.

Using the Multiplication Principle the number of ways of doing this is

$$C(n, r_1) \times C(n - r_1, r_2) \times \cdots \times C(n - r_1 - \cdots - r_{k-1}, r_k)$$

$$= \frac{n!}{r_1!(n-r_1)!} \times \frac{(n-r_1)!}{r_2!(n-r_1-r_2)!} \times \cdots \times \frac{(n-r_1-\cdots-r_{k-1})!}{r_k!0!} = \frac{n!}{r_1!r_2!\cdots r_k!}.$$

How many permutations of *Mississippi* are there?

Total number of letters with repeats is 11.

M occurs 1 time

i occurs 4 times

s occurs 4 times

p occurs 2 times

$$P(11;1,4,4,2) = \frac{11!}{1!\,4!\,4!\,2!}$$

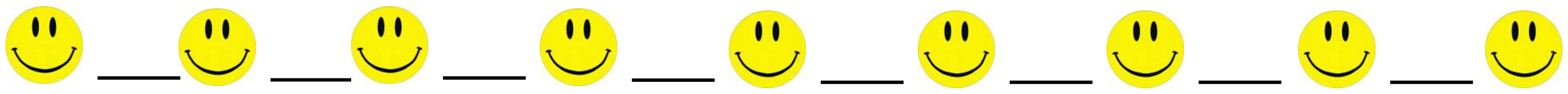
What has four eyes by can't see?

PSN. How many permutations of *engineering* are there?

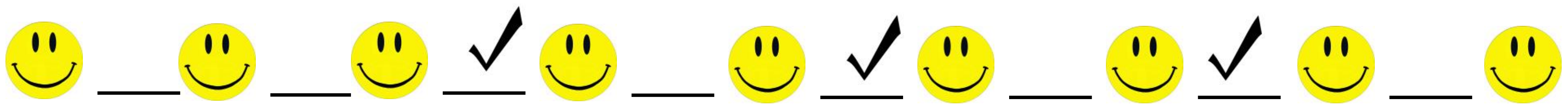
Combinations with Repetitions

Partitions of identical objects into *k* sets.

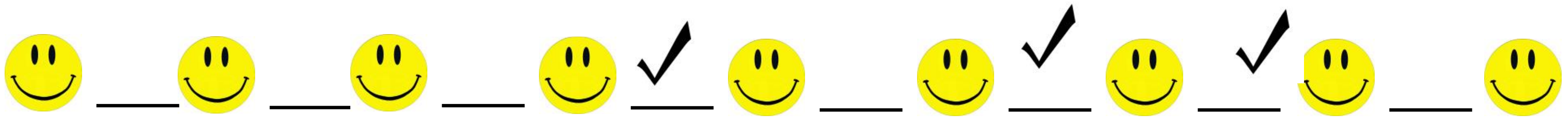
How many ways can we partition 9 smiley faces into 4 sets?



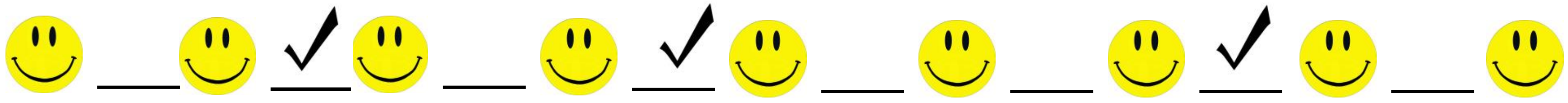
There are 8 slots between smiley faces. Check 3 of these slots to get partition into 3 sets of identical objects, i.e., smiley faces.



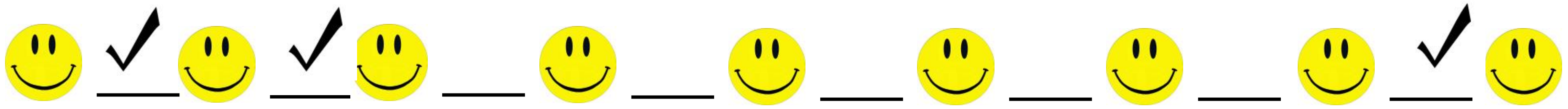
3 + 2 + 2 + 2



4 + 2 + 1 + 2



2 + 2 + 3 + 2



1 + 1 + 6 + 1

We are choosing 3 slots from a set of 8 slots. The number of ways of doing this is

$$C(8,3) = \frac{8!}{3!\,5!} = \frac{8 \times 7 \times 6}{3!} = 56$$

Note that this can be viewed as the number of integer solutions to

$$x_1 + x_2 + x_3 + x_4 = 9,$$
$$x_i \geq 1, \qquad i = 1, 2, 3, 4.$$

Theorem. The number of integer solution to the equation

$$x_1 + x_2 + \cdots + x_r = n,$$

$$x_i \geq 1, \qquad i = 1, \dots, r$$

is given by

$$C(n-1, r-1).$$

Proof

- Take n smiley faces with a slot between adjacent smiley faces.
- Then there are $n - 1$ slots.
- Check $r - 1$ of these slots.
- This partitions the smiley faces into r sets, giving a solution to the equation in the Theorem.
- The number of ways of choosing $r - 1$ slots from $n - 1$ to put the check in is $C(n - 1, r - 1)$. Q.E.D.

Corollary. The number of integer solution to the equation

$$x_1 + x_2 + \cdots + x_r = n,$$

$$x_i \geq 0, \qquad i = 1, \ldots, r$$

Is given by

$$C(n + r - 1, r - 1).$$

Proof of Corollary

Let $y_i = x_i + 1, i = 1, \ldots, r$.

Then, we have

$$y_1 + y_2 + \cdots + y_r = n + r - 1,$$
$$y_i \geq 1, \qquad i = 1, \ldots, r$$

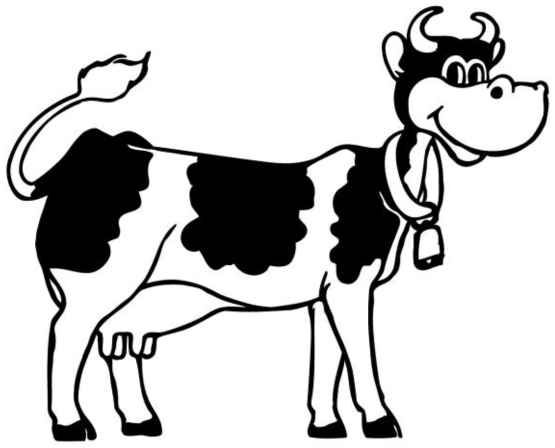
By the Theorem the number of solutions to the above is

$$C(n + r - 1, r - 1).$$

But this is the same as the number of solutions to the equation with the x_i's given in the Corollary.

How does a farmer count his cows?

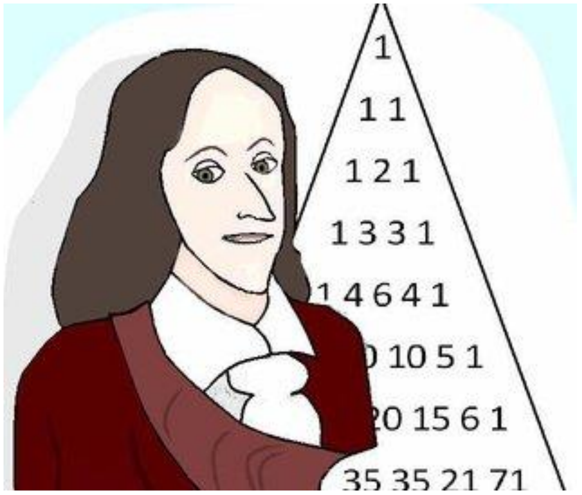
With a Cowculator.



Combinatorial Identities, Binomial Theorem

Textbook Reading:

Chapter 7. Sections 7.5, pp. 457-465



Combinatorial Identities

$$C(n,k) = C(n, n-k)$$

Newton's Identity

$$C(n,k) \times C(k,m) = C(n,m) \times C(n-m, k-m)$$

Pascal's Identity

$$C(n,k) = C(n-1,k) + C(n-1,k-1)$$

Proof of first identity

$$C(n,k) = \frac{n!}{k!\,(n-k)!} = \frac{n!}{(n-k)!\,k!} = C(n,n-k)$$

Combinatorial Proof of Newton's Identity

$$C(n,k) \times C(k,m) = C(n,m) \times C(n-m,k-m)$$

Consider the k-subsets of an n-element set where m elements in each k-subset are colored red. Using the Multiplication Principle this equals

$$C(n,k) \times C(k,m).$$

Another way of counting is to first take all the m-subsets of the n-element set, color them red, and add to each m-subset $k-m$ elements from the remaining $n-m$ elements to extend to a k-subset. Using the Multiplication Principle the number of ways of doing this is

$$C(n,m) \times C(n-m,k-m)$$

Algebraic Proof of Newton's Identity

$C(n,k) \times C(k,m) = C(n,m) \times C(n-m,k-m)$

$$C(n,k) \times C(k,m)$$

$$= \frac{n!}{k!(n-k)!} \times \frac{k!}{m!(k-m)!}$$

$$= \frac{n!}{k!(n-k)!} \times \frac{k!\,(n-m)!}{m!(k-m)!(n-m)!}$$

$$= \frac{n!}{m!(n-m)!} \times \frac{(n-m)!}{(k-m)!((n-m)-(k-m))!}$$

$$= C(n,m) \times C(n-m,k-m)$$

Algebraic Proof of Pascal's Identity

$$C(n,k) = C(n-1,k) + C(n-1,k-1)$$

$$C(n-1,k) + C(n-1,k-1)$$

$$= \frac{(n-1)!}{k!\,(n-1-k)!} + \frac{(n-1)!}{(k-1)!\,(n-k)!}$$

$$= \frac{(n-1)!\,(n-k)}{k!\,(n-k)!} + \frac{(n-1)!\,k}{k!\,(n-k)!}$$

$$= \frac{(n-1)!(n-k+k)}{k!(n-k)!} = \frac{n!}{k!(n-k)!} = C(n,k)$$

PSN. Give a combinatorial proof of Pascal’s Identity

$$C(n,k) = C(n-1,k) + C(n-1,k-1)$$

Binomial Theorem

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

The Binomial Theorem was first discovered by Sir Isaac Newton.

$C(n,k)$ is sometimes written as $\binom{n}{k}$ and sometimes referred to as a **binomial coefficient**.

Proof for $n = 4$

$$(x+y)(x+y)(x+y)(x+y)$$

Using the distributive law the

coefficient of x^4 is obtained by choosing x in all terms: $C(4,4)$ ways

coefficient of x^3y is obtained by choosing x is 3 terms: $C(4,3)$ ways

coefficient of $x^2y^2$is obtained by choosing x is 2 terms: $C(4,2)$ ways

coefficient of $xy^3$is obtained by choosing x is 1 term: $C(4,1)$ ways

coefficient of y^4 is obtained by choosing x is 0 terms: $C(4,0)$ ways

Combining we have

$$(x+y)^4 = (x+y)(x+y)(x+y)(x+y)$$

$$= C(4,4)x^4 + C(4,3)x^3y + C(4,2)x^2y^2 + C(4,1)xy^3 + C(4,0)y^4$$

Proof for general n

$$(x+y)(x+y)\cdots(x+y)(x+y)$$

Using the distributive law the

coefficient of x^n is obtained by choosing x in all terms: $C(n,n)$ ways

coefficient of $x^{n-1}y$ is obtained by choosing x is 3 terms: $C(n,n-1)$ ways

coefficient of $x^{n-2}y^2$is obtained by choosing x is 2 terms: $C(n,n-2)$ ways

$$\vdots$$

coefficient of $xy^{n-1}$is obtained by choosing x is 1 term: $C(n,1)$ ways

coefficient of y^n is obtained by choosing x is 0 terms: $C(n,0)$ ways

Combining we have

$$(x+y)^n = (x+y)(x+y)\cdots(x+y)(x+y)$$

$$= C(n,n)x^n + C(n,n-1)x^{n-1}y + C(n,n-2)x^{n-2}y^2 + C(n,1)xy^{n-1} + C(n,0)y^n$$

$$= \sum_{i=0}^{n} C(n,i)x^{n-i}y^i$$

Pascal’s Triangle

```
                              1
                           1     1
                        1     2     1
                     1     3     3     1
                  1     4     6     4     1
               1     5    10    10     5     1
            1     6    15    20    15     6     1
         1     7    21    35    35    21     7     1
      1     8    28    56    70    56    28     8     1
   1     9    36    84   126   126    84    36     9     1
1    10    45   120   200   252   200   120    45    10     1
```

Sum of Row in Pascal's Triangle

Show that the n^{th} row of Pascal's triangle sums to 2^n.

We need to show that

$$C(n,0) + C(n,1) + \cdots + C(n,n) = 2^n$$

By the binomial theorem

$$(1+1)^n = \sum_{i=0}^{n} C(n,i) 1^{n-i} 1^i = \sum_{i=0}^{n} C(n,i)$$

Combinatorial Proof

Let $S = \{1,2,\dots,n\}$. The cardinality of the power set is given by

$$|P(S)| = 2^n$$

Let S_k denote the k-subsets, $k = 0,1,\dots,n$. Clearly,

$$|P(S)| = |S_0| + |S_1| + \cdots + |S_n|$$

Then

$$|S_k| = C(n,k)$$

Substituting we obtain

$$2^n = |P(S)| = |S_0| + |S_1| + \cdots + |S_n|$$

$$= C(n,0) + C(n,1) + \cdots + C(n,n)$$

PSN

Show that

$$C(n,0) - C(n,1) + \cdots + (-1)^n C(n,n) = 0$$

Multinomial Coefficient

Recall that if we k symbols distinct symbols, where the i^{th} symbol has multiplicity $r_i, i = 1, \dots, k$ and the total number of symbols with repetition is $n = r_1 + r_2 + \cdots + r_k$, then the number of permutations with repetitions is given by

$$P(n; r_1, r_2, \dots, r_k) = \frac{n!}{r_1! r_2! \cdots r_k!}.$$

This can be referred to a multinomial coefficient and denoted by

$$\binom{n}{r_1 \; r_2 \; \cdots \; r_k}$$

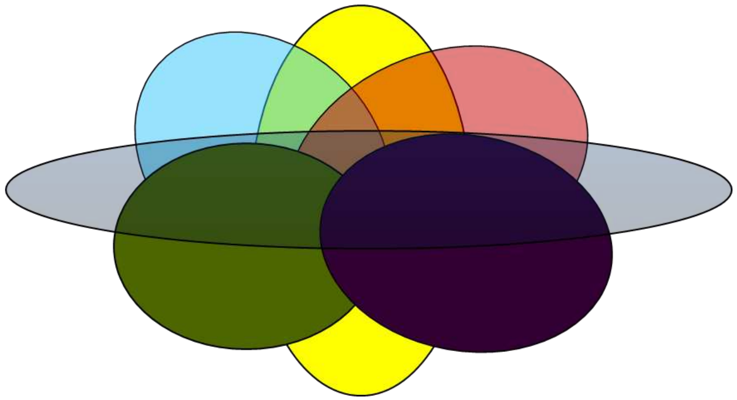
Multinomial Theorem

$$(x_1 + x_2 + \cdots x_k)^n = \sum_{r_1+r_2+\cdots r_k=n} \binom{n!}{r_1 \;\; r_2 \;\; \ldots \;\; r_k} {x_1}^{r_1} \, {x_2}^{r_2} \ldots {x_k}^{r_k}$$

Why didn't Isaac Newton dodge the apple?

He didn't understand the gravity of the situation.

Proof of Principle of Inclusion-Exclusion and Counting Derangements



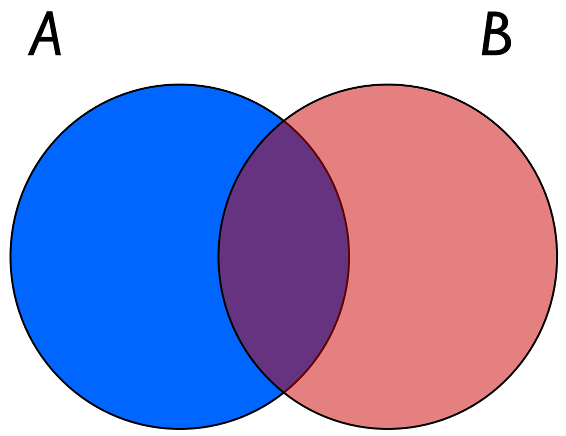
Textbook Reading:

Section 7.5, pp. 445-446

Principle of Inclusion-Exclusion (2 sets)

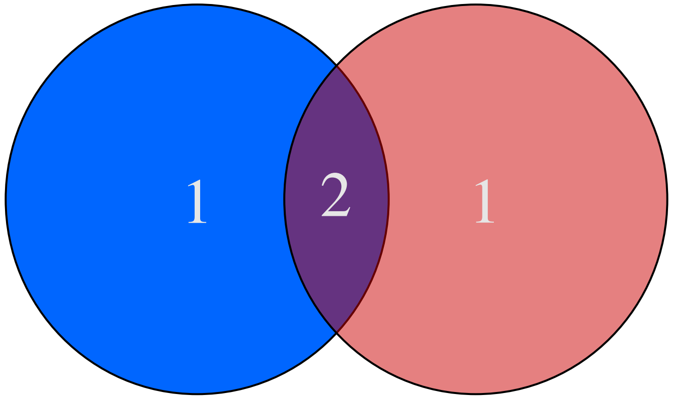
For two arbitrary sets *A* and *B*

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

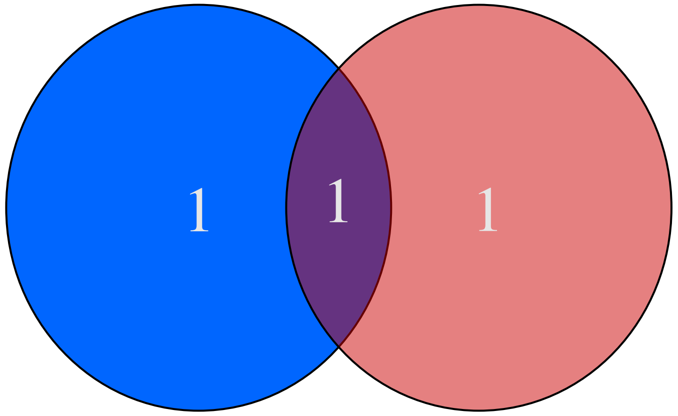


Proof of Principle of Inclusion-Exclusion (2 sets)

$|A| + |B|$

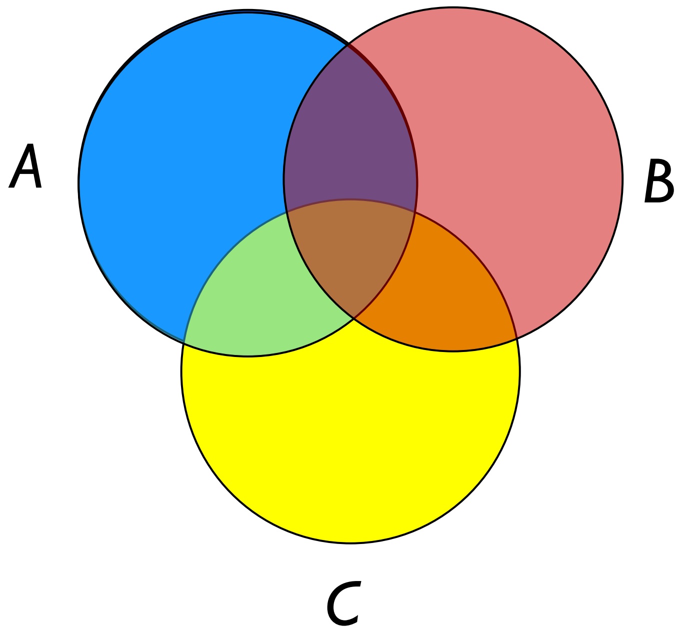


$|A| + |B| - |A \cap B|$



$|A \cup B|$

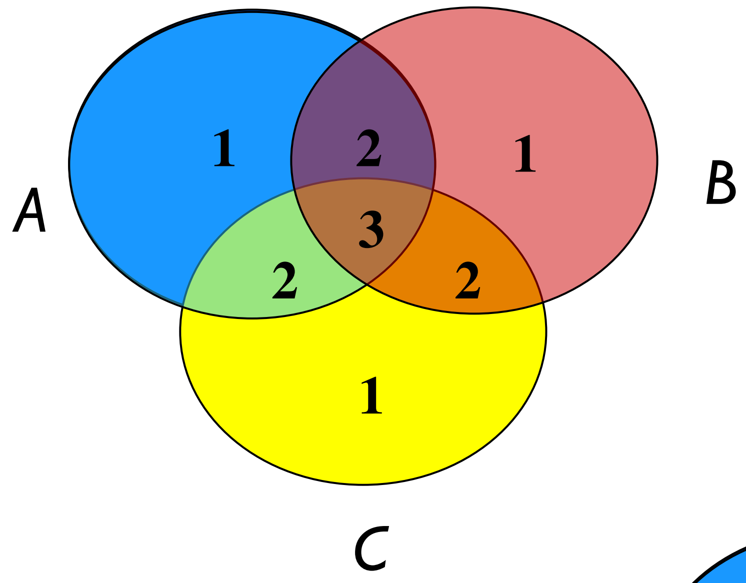
Principle of Inclusion-Exclusion (3 sets)



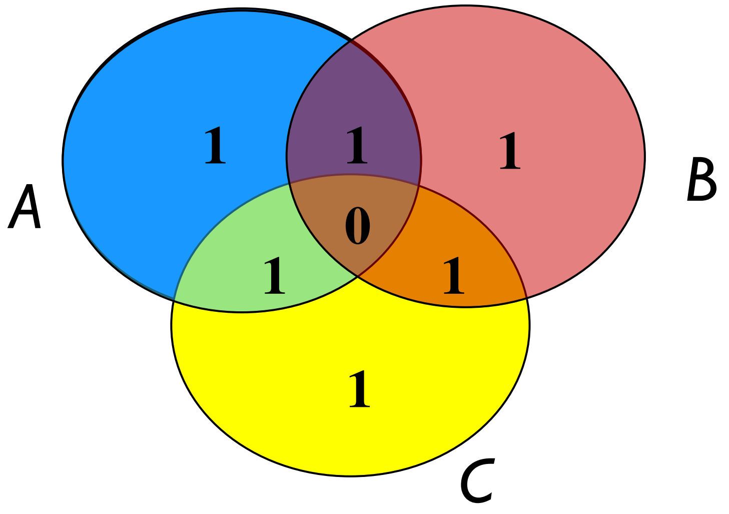
$$\boxed{\begin{aligned}|A \cup B \cup C| = {} & |A| + |B| + |C| \\ & - |A \cap B| - |A \cap C| - |B \cap C| \\ & + |A \cap B \cap C|\end{aligned}}$$

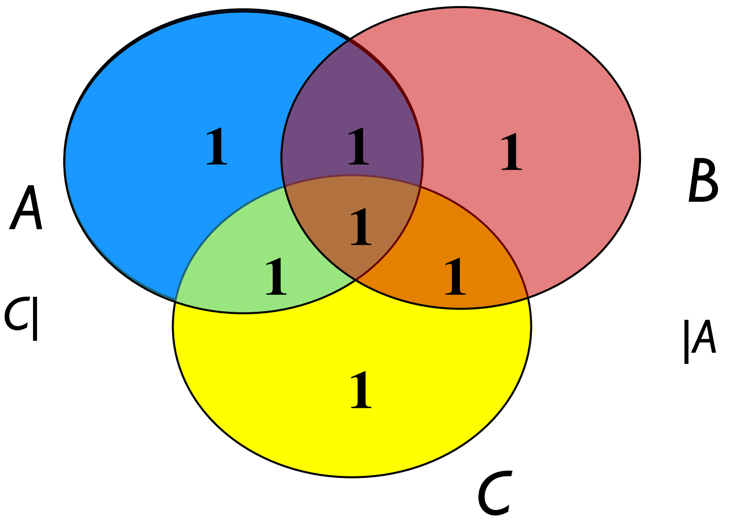
Proof of Principle of Inclusion-Exclusion (3 sets)

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



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

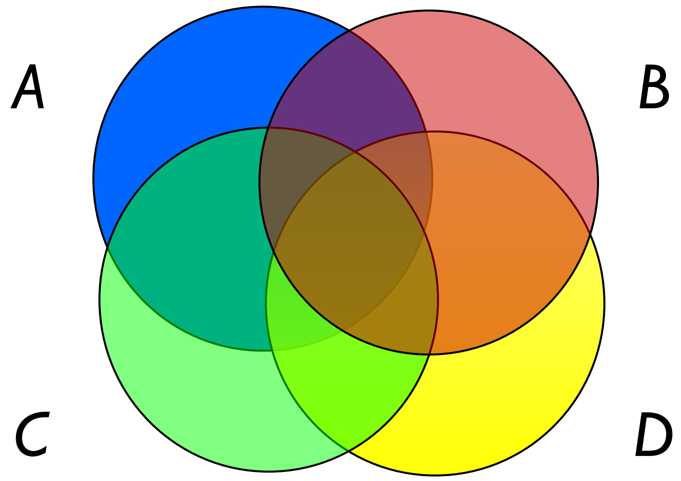




$|A| + |B| + |C|$
$- |A \cap B| - |A \cap C| - |B \cap C|$
$+ |A \cap B \cap C|$

$|A \cup B \cup C|$

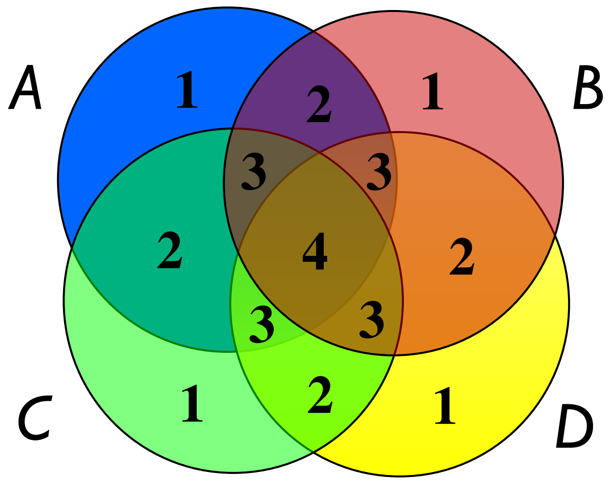
Principle of Inclusion-Exclusion (4 sets)



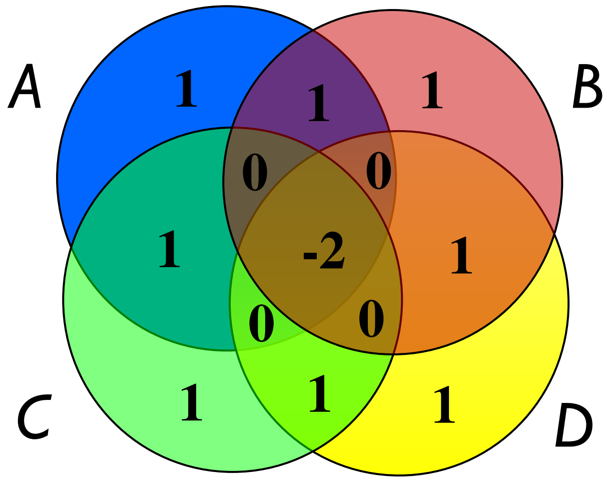
$$
\begin{aligned}
|A \cup B \cup C \cup D| = {} & |A| + |B| + |C| + |D| \\
& - |A \cap B| - |A \cap C| - |A \cap D| - |B \cap C| - |B \cap D| - |C \cap D| \\
& + |A \cap B \cap C| + |A \cap B \cap D| + |A \cap C \cap D| + |B \cap C \cap D| \\
& - |A \cap B \cap C \cap D|
\end{aligned}
$$

Proof of Principle of Inclusion-Exclusion (4 sets)

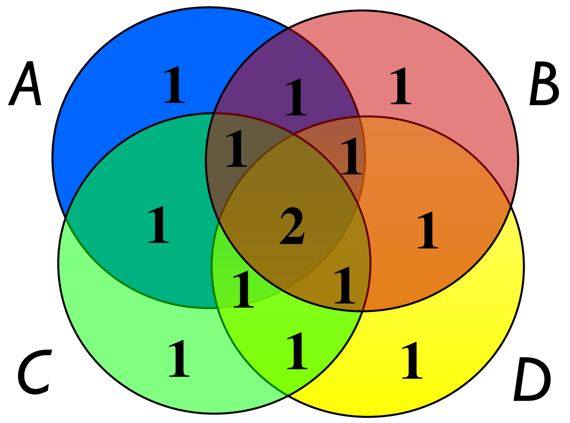
|A| + |B| + |C| + |D|



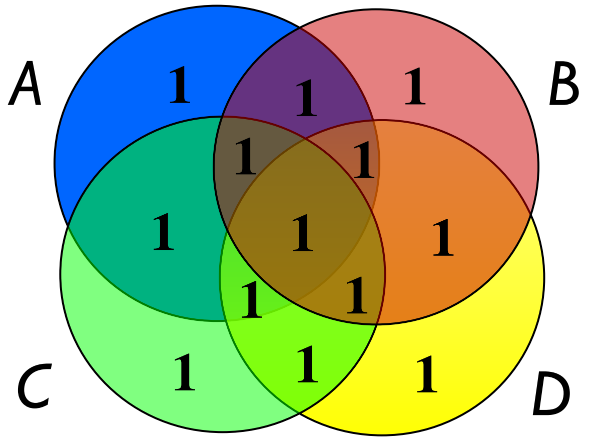
|A| + |B| + |C| + |D|
– |A ∩ B| – |A ∩ C| – |A ∩ D| – |B ∩ C| – |B ∩ D| – |C ∩ D|



|A| + |B| + |C| + |D|
– |A ∩ B| – |A ∩ C| – |A ∩ D| – |B ∩ C| – |B ∩ D| – |C ∩ D|
+ |A ∩ B ∩ C| + |A ∩ B ∩ D| + |A ∩ C ∩ D| + |B ∩ C ∩ D|

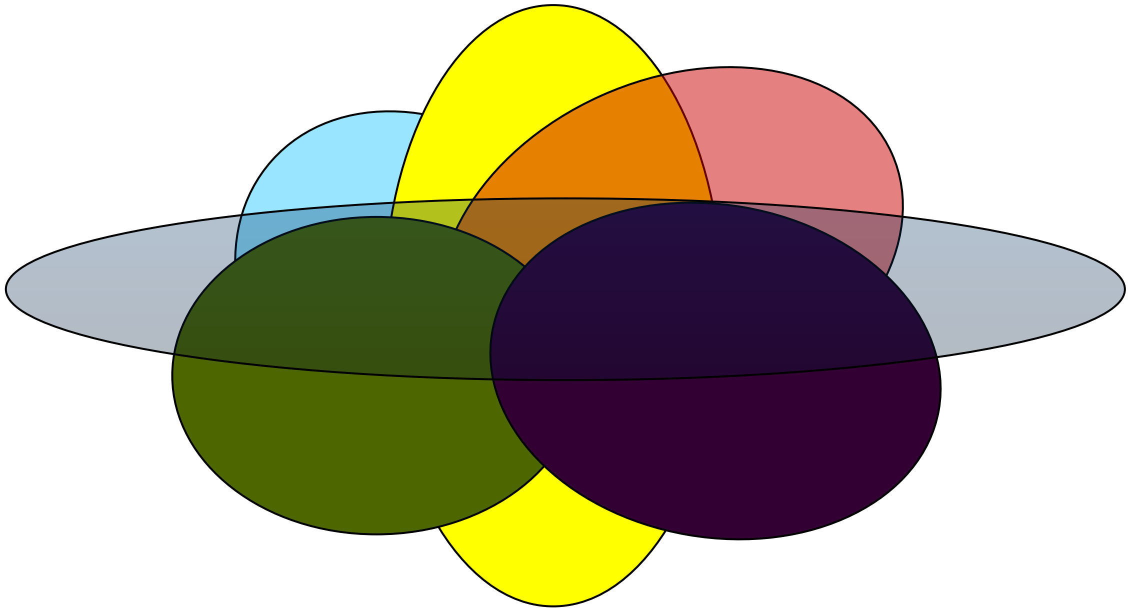


|A| + |B| + |C| + |D|
– |A ∩ B| – |A ∩ C| – |A ∩ D| – |B ∩ C| – |B ∩ D| – |C ∩ D|
+ |A ∩ B ∩ C| + |A ∩ B ∩ D| + |A ∩ C ∩ D| + |B ∩ C ∩ D|
– |A ∩ B ∩ C ∩ D | = |A ∪ B ∪ C ∪ D|



Principle of Inclusion-Exclusion (*n* sets)

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



Principle of 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

⋮

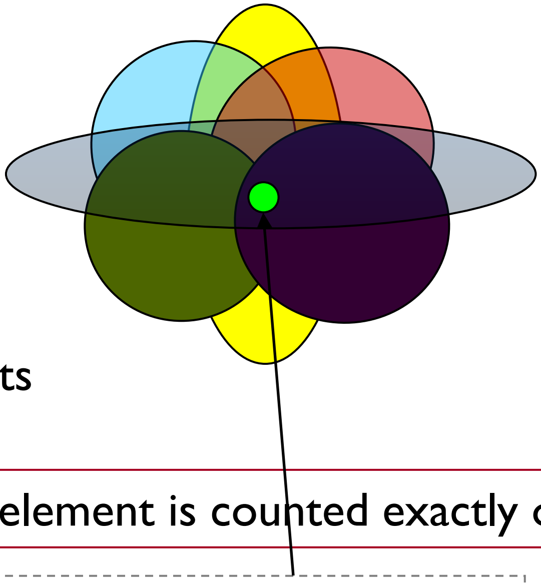
+ $(-1)^{n+1} \times$ sum of sizes of intersections of all *n* sets

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

Proof of Principle of 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



To prove correctness we need 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:

$$C(k,1) - C(k,2) + C(k,3) - C(k,4) + \cdots + (-1)^{k+1}C(k,k)$$

It remains to show this sum equals 1.

Proof cont'd

By the Binomial Theorem

$$0 = (1-1)^k = \sum_{i=0}^{k} C(k,i)1^{k-i}(-1)^i = \sum_{i=0}^{k}(-1)^i C(k,i)$$

$$= C(k,0) - C(k,1) + C(k,2) - C(k,3) + \cdots + (-1)^k C(k,k)$$

Using fact that $C(k,0) = 1$, we have

$$\Rightarrow C(k,1) - C(k,2) + C(k,3) - \cdots + (-1)^{k+1} C(k,k) = 1$$

This completes the proof of the Principle of Inclusion-Exclusion.

Derangements

A **fixed point** of a permutation π is an index i such that $\pi(i) = i$.

A **derangement** is a permutation with **no** fixed points.

PSN.

a) Compute all derangements for $n = 2, 3, 4$.

b) Compute the fraction of permutations that are derangements for $n = 2, 3, 4$.

Computing number of derangements for *n* = 4

- We will use complement and the Principle of Inclusion-exclusion.
- Let P_4 denote the set of all permutations on $n = 4$ elements
- Let D_4 denote the set of all derangements on $n = 4$ elements
- Let d_4 denote the set of all derangements on $n = 4$ elements
- Let A denote the set of all permutations from P_4 that have a fixed point in at least one location.
- Then $d_4 = |D_4| = |P_4| - |A| = 4! - |A|$

Computing number of derangements for *n* = 4

- Let A_i denote the number of permutations on 4 elements where there is a fixed point in position $i, i = 1,2,3,4.$
- Then $A = A_1 \cup A_2 \cup A_3 \cup A_4$
- So using result from previous slide we have

$$d_4 = 4! - |A| = 4! - |A_1 \cup A_2 \cup A_3 \cup A_4|$$

Computing number of derangements for *n* = 4

Placing i in position i there are 3! ways of placing the remaining elements, so

$$|A_i| = 3!$$

Placing i in position i and j in position j there are 2! ways of placing the remaining elements, so

$$|A_i \cap A_j| = 2!$$

Placing i in position i, j in position j, and k in position k there are 1! ways of placing the remaining elements, so

$$|A_i \cap A_j \cap A_k| = 1!$$

Finally,

$$|A_1 \cap A_2 \cap A_3 \cap A_4| = 0!$$

Applying Principle of Inclusion-Exclusion

$$|A_1 \cup A_2 \cup A_3 \cup A_4|$$

$$= |A_1| + |A_2| + |A_3| + |A_4|$$
$$- |A_1 \cap A_2| - |A_1 \cap A_3| - |A_1 \cap A_4| - |A_2 \cap A_3| - |A_2 \cap A_4| - |A_3 \cap A_4|$$
$$+ |A_1 \cap A_2 \cap A_3| + |A_1 \cap A_2 \cap A_4| + |A_1 \cap A_3 \cap A_4| + |A_2 \cap A_3 \cap A_4|$$
$$- |A_1 \cap A_2 \cap A_3 \cap A_4|$$

$$= C(4,1)3! - C(4,2)2! + C(4,3)1! - C(4,4)0!$$

Using result from previous slide and substituting, we have

$$d_4 = 4! - |A| = 4! - |A_1 \cup A_2 \cup A_3 \cup A_4|$$

$$= 4! - C(4,1)3! + C(4,2)2! - C(4,3)1! + C(4,4)0!$$

$$= 4! - \frac{4!}{1!3!}3! + \frac{4!}{2!2!}2! - \frac{4!}{3!1!}1! + \frac{4!}{4!0!}0!$$

$$= 4! - \frac{4!}{1!} + \frac{4!}{2!} - \frac{4!}{3!} + \frac{4!}{4!} = 4!\,(1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + \frac{1}{4!})$$

Thus,

$$\frac{d_4}{4!} = 1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + \frac{1}{4!} = .375$$

This is the same result we obtained on the previous slide where we used brute force, i.e., we enumerated all the permutations and derangements of a 4-element set.

Can you conjecture what the formula is for a general *n*?

Computing number of derangements

- Let A_i denote the number of permutations on n elements where there is a fixed point in position $i, i = 1,2, \ldots, n$.
- Then $A = A_1 \cup A_2 \cup \cdots \cup A_n$
- So using result from previous slide we have

$$d_n = n! - |A| = n! - |A_1 \cup A_2 \cup \cdots \cup A_n|$$

Cardinalities of intersection of sets A_i

Placing i in position i there are $(n-1)!$ ways of placing the remaining elements, so

$$|A_i| = (n-1)!$$

Placing i position i and j in position j there are $(n-2)!$ ways of placing the remaining elements, so

$$|A_i \cap A_j| = (n-2)!$$

Placing i position i, j in position j, and k in position k there are $(n-3)!$ ways of placing the remaining elements, so

$$|A_i \cap A_j \cap A_k| = (n-3)!$$

$$\vdots$$

$$|A_1 \cap A_2 \cap \cdots \cap A_n| = 0!$$

Applying Principle of Inclusion-Exclusion

$$
\begin{aligned}
& |A_1 \cup A_2 \cup A_3 \cup \cdots \cup A_n| \\
= \ & |A_1| + |A_2| + |A_3| + |A_4| \\
& - |A_1 \cap A_2| - |A_1 \cap A_3| - |A_1 \cap A_4| - |A_2 \cap A_3| - |A_2 \cap A_4| - |A_3 \cap A_4| - \cdots \\
& + |A_1 \cap A_2 \cap A_3| + |A_1 \cap A_2 \cap A_4| + |A_1 \cap A_3 \cap A_4| + |A_2 \cap A_3 \cap A_4| + \cdots \\
& \quad \vdots \\
& +(-1)^n |A_1 \cap A_2 \cap \cdots \cap A_n| \\
= \ & C(n,1)(n-1)! - C(n,2)(n-2)! + C(n,3)(n-3)! - \cdots + (-1)^n C(n,n)0!
\end{aligned}
$$

Using result from previous slide and substituting, we have

$$
\begin{aligned}
d_n &= n! - |A| = n! - |A_1 \cup A_2 \cup \cdots \cup A_n| \\
&= n! - C(n,1)(n-1)! + C(n,2)(n-2)! - C(n,3)(n-3)! + \cdots + (-1)^n C(4,4)0! \\
&= n! - \frac{n!}{1!(n-1)!}(n-1)! + \frac{n!}{2!(n-2)!}(n-2)! - \frac{n!}{3!(n-3)!}(n-3)! + \cdots + (-1)^n \frac{n!}{n!0!}0! \\
&= n! - \frac{n!}{1!} + \frac{n!}{2!} - \frac{n!}{3!} + \cdots + (-1)^n \frac{n!}{n!} = n!\left(1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + \cdots + (-1)^n \frac{1}{n!}\right)
\end{aligned}
$$

Conclusion

Thus,

$$\frac{d_n}{n!} = 1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + \cdots + (-1)^n \frac{1}{n!}$$

Can you conjecture what the limit equals as $n \to \infty$?

Probability of choosing a derangement

$$\lim_{n\to\infty} \frac{d_n}{n!} = \lim_{n\to\infty} \left(1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + \cdots + (-1)^n \frac{1}{n!}\right)$$

$$= e^{-1}$$

$$\approx .367879441$$

Thus, for large n, the fraction of permutations that are derangements, or equivalently the probability of choosing a permutation that it is a derangement, is about $\frac{1}{e} \approx .367879441$

Probability Theory

Textbook Reading:

Chapter 8, pp.475-498

Applications of Probability Theory

Statistics

Inferences on statistics on data collected
are made under the framework
of probability theory.

Applications of Probability Theory, cont’d

Probabilistic Analysis of an Algorithm

The expected performance of an algorithm for inputs of a given size is referred to as the average complexity and is an important measure of computing time.

Applications of Probability Theory, cont’d

Probabilistic Algorithms

- Most algorithms are deterministic. A deterministic algorithm performs exactly the same for every run with the same input
- On the other hand, a **probabilistic** algorithm contains steps that make random choices by invoking a random (or pseudorandom) number generator.
- Thus, they are subject to the laws of chance. In particular, a probabilistic algorithm can perform differently for two runs with the same input.
- The advantage of using a probabilistic algorithm is to obtained better expected computing time in the worst case.

Discrete Sample Space

- A **discrete sample space** *S* is a nonempty set that has only a finite or countably infinite number of elements.
- An element of *S* is called an **outcome.**
- A subset of *S* is called an **event**.

Probability Density Function

A **probability density function** p on a discrete sample space S is defined by

$$\text{For each } s\epsilon S, \quad 0 \leq p(s) \leq 1$$

and

$$\sum_{s\in S} p(s) = 1$$

These two conditions determine the two axioms of probability theory.

A probability density function is also called a **probability mass function**.

Probability of an outcome and event

- The value *p*(*s*) of the probability density function of outcome *s* is the **probability of *s***.
- Let *E* be an event in a sample space S with a probability density function *p*(*s*).
- The probability *P*(*E*) of event *E* is the sum of the probabilities over all the outcomes *E* , i.e.,

$$P(E) = \sum_{s \in E} p(s)$$

Example: Rolling a die

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

Outcomes are 1, 2, 3, 4, 5, 6. If die is fair

$$p(i) = \frac{1}{6}, i = 1, 2, \dots, 6.$$

Let E be the event value is odd, i.e.,

$$E = \{1,3,5\}$$

$$P(E) = p(1) + p(3) + p(5) = \frac{1}{6} + \frac{1}{6} + \frac{1}{6} = \frac{1}{2}.$$

Example: Rolling 2 dice

$$S = \{(i,j) : i,j = 1,2,\ldots,6\}$$

Outcomes are $(i,j) : i,j = 1,2,\ldots,6$. If both dice are fair

$$P((i,j)) = \frac{1}{36}, i,j = 1,2,\ldots,6.$$

Let E be the event sum of dice is seven, i.e.,

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

$$P(E) = p(1,6) + p(2,5) + p(3,4) + p(4,3) + p(5,2) + p(6,1)$$

$$= \frac{1}{36} + \frac{1}{36} + \frac{1}{36} + \frac{1}{36} + \frac{1}{36} + \frac{1}{36} = \frac{1}{6}.$$

Cartesian Product of Sample Spaces

Let $S_1, S_2, \dots, S_n$ be respectively. Then the sample space S that is their cartesian product is given by

$$S = S_1 \times S_2 \times \cdots \times S_n$$

$$= \{(s_1, s_2, \dots, s_n) : s_i \in S_i, i = 1, 2, \dots, n\}$$

The probability of an outcome $s = (s_1, s_2, \dots, s_n)$ is given by

$$p(s) = p(s_1) \times p(s_2) \times \cdots \times p(s_n)$$

Example with two dice

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

$$S \times S = \{(i,j) : i,j = 1, 2, \dots, 6\}$$

$$p(i,j) = p(i) \times p(j) = \frac{1}{6} \times \frac{1}{6} = \frac{1}{36}.$$

Probability of Events in Cartesian Product of Sample Spaces

Let $E_1, E_2, \ldots, E_n$ be events in sample spaces $S_1, S_2, \ldots, S_n$, respectively. Then,

$$P(E_1 \times E_2 \times \cdots \times E_n)$$

$$= P(E_1)P(E_2) \ldots P(E_n)$$

The Frequency Interpretation of Probability

- The frequency interpretation of probability is to take the quantity $P(E)$ as an estimate for the proportion of times that event E will occur when an experiment is repeated over and over.
- The reasonableness of the estimate depends on how well the probability density function estimates the frequencies of the outcomes

Uniform Probability Density Function

A probability density function is **uniform** if every outcome has the same probability.

The common probability is $\frac{1}{|S|}$. Thus,

$$p(s) = \frac{1}{|S|}, \forall s \in S$$

The probability of an event E is given by

$$P(E) = \sum_{s \in E} p(s) = \sum_{s \in E} \frac{1}{|S|} = \frac{|E|}{|S|}$$

PSN. Drawing from a Deck of Cards.

Assuming a uniform probability density function on the standard deck of cards.

a) Determine the probability of drawing an Ace.

b) Determine the probability of drawing a face card, i.e., Jack, Queen, or King.

Poker Hands

The sample space S is the set of all 5-card hands

$$|S| = C(52,5) = \frac{52!}{5!\,48!} = \frac{52 \times 51 \times 50 \times 49 \times 48}{5!} = 2{,}598{,}960$$

It follows that for $s \in S$,

$$p(s) = \frac{1}{2{,}598{,}960}.$$

Let *H* an event, i.e., a set of hands, such as a pair, two pair, full house, etc. For a uniform probability density function. Then,

$$P(H) = \frac{|H|}{|S|} = \frac{|H|}{C(52,5)}$$

Probability of poker hands

The probability of a hand with one pair is

$$\frac{C(13,1) \times C(4,2) \times C(12,3) \times C(4,1) \times C(4,1) \times C(4,1)}{C(52,5)}$$

The probability of a hand with two pairs is

$$\frac{C(13,2) \times C(4,2) \times C(4,2) \times C(11,1) \times C(4,1)}{C(52,5)}$$

The probability of a full house is

$$\frac{C(13,1) \times C(4,2) \times C(12,1) \times C(4,3)}{C(52,5)}$$

Disjoint events

Let $E_1, E_2, \dots, E_n$ be pairwise disjoint events. Then,

$$P(E_1 \cup E_2 \cup \cdots \cup E_n) = P(E_1) + P(E_2) + \cdots + P(E_n)$$

Properties of probability of events

Let E and F be event in a sample space S and let $\bar{F}$ denote the complement of F, *i.e.*, $\bar{F} = S - F$

a) $E \subseteq F \Rightarrow P(E) \leq P(F)$

b) $P(E \cup F) = P(E) + P(F) - P(E \cap F)$

c) $P(E) = P(E \cap F) + P(E \cap \bar{F})$

The last property can be generalized to *n* events $F_1, F_2, \ldots, F_n$ partition S:

$$P(E) = P(E \cap F_1) + P(E \cap F_2) + \cdots + P(E \cap F_n)$$

PSN. What is the probability that a card drawn at random from a 52-card deck will be an King or a heart?

Probability of the complement

Let E be event in a sample space S and let $\bar{E}$ denote the complement of F, i.e., $\bar{E} = S - F$. Then,

$$P(E) = 1 - P(\bar{E})$$

The Birthday Problem

What is the probability that in a group of *n* people, at least two have the same birthday?

Leap years are ignored, and all combinations of birthdays are assumed to be equally likely.

Solution to the Birthday Problem

- Represent the birthdays of the group by an n-tuple with components that are integers in the range 1 through 365.
- Take the sample space S to the 365^n possible n-tuples, and assume the
- Assume a uniform probability density on S.
- Let E be the event that at least two people have the same birthday.
- Then $\bar{E}$ is the event that no two people have the same birthday
- Then

$$P(E) = 1 - P(\bar{E})$$

Computing $\bar{E}$

- $\bar{E}$ consists of all possible n-tuples of distinct birthdays
- The number of elements in $\bar{E}$ can be counted as follows:
  - the first person has any of 365 birthdays
  - the second has any of the 364 remaining possibilities

    $\vdots$

  - the nth person has any of the remaining $365 - n + 1$ possibilities
- Thus,

$$|\bar{E}| = 365 \times 364 \times \cdots \times (365 - n + 1)$$

Computing $P(E)$

Since the probability density function is uniform

$$P(\bar{E}) = \frac{|\bar{E}|}{|S|} = \frac{365 \times 364 \times \cdots \times (365-n+1)}{365^n}.$$

Therefore, we have

$$P(E) = 1 - P(\bar{E}) = 1 - \frac{365 \times 364 \times \cdots \times (365-n+1)}{365^n}$$

Interesting Conclusion

Using a calculator and the identity

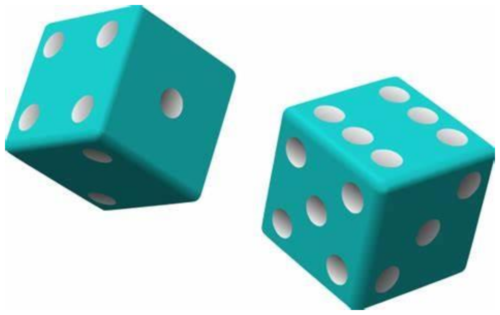
$$P(\bar{E}) = \frac{365 \times 364 \times \cdots \times (365 - n + 1)}{365^n}$$

$$= \frac{365}{365} \times \frac{364}{365} \times \cdots \times \frac{365 - n + 1}{365}$$

it is easy to compute that for $n \geq 23$, $P(\bar{E}) < 0.5$ and $P(E) > 0.5$.

In other words, if at least 23 students are in a class, then the chances are greater than 50% that some two or more of them have the same birthday.

Why do dice always prefer to be in twos?



Because it's a pair a' dice.

Discrete Probability: Bernoulli Process, Conditional Probability, Random Variable

Textbook Reading:

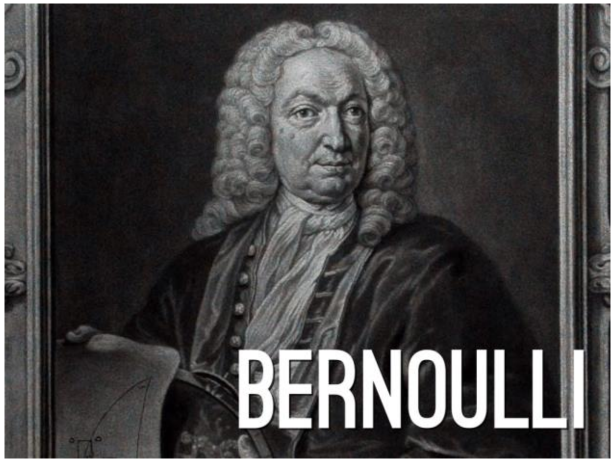
Chapter 8, pp. 498-521

Bernoulli Process

A **Bernoulli trial process** or simply a **Bernoulli process** is a sequence of repetitions, called trials, of an experiment with a two-element sample space.

It is assumed that the trials have no influence on one another.

The two possible outcomes of a trial need not be equally likely.



n-trial Bernoulli Process

In an n-trial Bernoulli process the trial is repeated n times with probability p of success on each trial. The probability of failure is $q = 1 - p$.

For example a trial could be flipping a coin. If success is a head (and failure at tail) then

$$p = q = \frac{1}{2}.$$

Another example is rolling a die. If the die is fair and success is a 5 or 6 (and failure a 1, 2, 3 or 4) then

$$p = \frac{2}{6} = \frac{1}{3} \text{ and } q = \frac{2}{3}.$$

Probability of *k* successes in a Bernoulli Process

- Let $b(n; k, p)$ denote the probability of exactly k successes in an n-trial Bernoulli process with probability p of success on each trial.
- The number of ways of choosing k trials (from trials $1, 2, \ldots, n$) to be successes is

$$C(n, k)$$

- The probability that a given set of k trials are successes and the remaining $n - k$ are failures is

$$p^k q^{n-k}$$

- It follows that

$$b(n; k, p) = C(n, k) p^k q^{n-k}$$

Sample Space associated with Bernoulli Process

Let $S = \{s_1,…, s_n\}$ where outcome s_i means exactly i successes. Then, $p(s_i) = b(n; i, p) = C(n, i)p^i q^{n-i}$

determines a probability density function on S.

Proof. Applying the Binomial Theorem we obtain

$$\sum_{i=1}^{n} p(s_i) = \sum_{i=1}^{n} C(n, i)p^i q^{n-i} = (p + q)^n = 1^n = 1$$

Independent Pairs of Events

A pair of events A and B belonging to the same sample space are said to be **independent** if

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

PSN

Consider one roll of a fair die with $S = \{1, 2, 3, 4, 5, 6\}$

Let A be the event that an odd number is rolled, i.e.,

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

Let B be the event that the number rolled is at least three, i.e.,

$$B = \{3, 4, 5, 6\}$$

Let C be the event that the number rolled is a prime, i.e.,

$$C = \{2, 3, 5\}$$

Which pairs of events are independent?

Independent Set of Events

A set of events $\{A_1, A_2, \ldots, A_n\}$ from sample space S is an **independent set of events** if

$$P(A_1 \cap A_2 \cdots \cap A_n) = P(A_1)P(A_2)\cdots P(A_n)$$

Conditional Probability

- Conditional probabilities often play a role when we analyze the average behavior of algorithms.
- To motivate the notion of conditional probability, consider again the experiment of rolling two fair dice D_1 and D_2.
- Suppose that you have been informed that the first die came up 2. You now are asked to determine the probability that the sum of the two dice is at most 5.
- This is the *conditional probability* that the sum of the dice is at most 5, *given* that the first die has come up 2.
- Let E denote the event that the sum of the two dice is at most 5, i.e., $E = \{(1,1),(1,2),(1,3),(1,4),(2,1),(2,2),(2,3),(3,1),(3,2),(4,1)\}$ and let F denote the event that the first die is a 2, i.e., $F = \{(2,1),(2,2),(2,3),(2,4),(2,5),(2,6)\}$
- E contains 10 outcomes and F contains 6 outcomes, so that $P(E) = 10/36 = 5/18$ and $P(F) = 6/36 = 1/6$.
- Let $P(E|F)$ denote the conditional probability that E occurs given that F has occurred.
- The condition that D_1 comes up 2 amounts to restricting the sample space to $F = \{(2,i)|i \in \{1, . . . , 6\}\}$,
- This restricted sample space has 6 outcomes.
- Of these 6 outcomes only the 3 in $E \cap F = \{(2,1), (2,2), (2,3)\}$ yield a sum that is at most 5. Thus,

$$P(E \mid F) = \frac{|E \cap F|}{|F|} = \frac{3}{6} = \frac{1}{2}.$$

Formula for Conditional Probility

Since $\frac{|E \cap F|}{|F|} = \frac{|E \cap F|/|S|}{|F|/|S|} = \frac{P(E \cap F)}{P(F)}$, we may express $P(E|F)$ in terms of the probabilities of events $E \cap F$ and F

$$P(E|F) = \frac{P(E \cap F)}{P(F)}$$

Substituting $P(E \cap F) = \frac{3}{36} = \frac{1}{12}$ and $P(F) = \frac{1}{6}$ yields $P(E|F) = \frac{1}{2}$ the same result as on the previous slide.

PSN. Compute the conditional probability that the sum of dice is 2 or 11 given that the first die is a 1 or 6.

Total Probability Theorem

Let $E_1, E_2, \ldots, E_n$ be events that partition S and let A be any event in S. Then

$$P(A) = P(A|E_1)P(E_1) + P(A|E_2)P(E_2) + \cdots + P(A|E_n)P(E_n)$$

Bayes Rule

Let A and B be events in sample space S. If neither $P(A)$ nor $P(B)$ is zero, then

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}$$

Example

- Suppose it is estimated that 10% of a population has a certain disease.
- Tests for this disease are being developed but are not yet perfect. In fact, an individual who has the disease may test negative.
- Suppose experience with a particular test shows that 5% of the results are actually false negatives-that is, the individual actually does have the disease.
- Also, suppose that 8% of the tests done so far have been positive.
- What is the probability that a sick person will receive a false-negative test result?

Solution

Let S consist of the population.

Let A denote the subset of people who would test positive if they took the test.

Let B denote the subset of people who have the disease.

We are given that

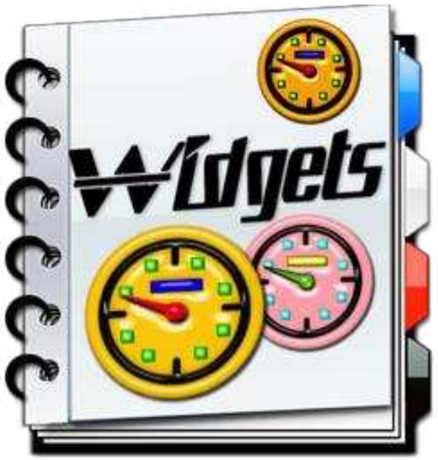
$$P(A) = 0.08\ , P(B) = 0.10,\ P(B|\bar{A}) = 0.05$$

Then $P(\bar{A}) = 1 - P(A) = 0.92$. We are interested in the probability that a person who is ill tests negative, i.e., $P(\bar{A}|B)$. By Bayes' Rule

$$P(\bar{A}|B) = \frac{P(B|\bar{A})P(\bar{A})}{P(B)} = \frac{(0.05)(0.92)}{0.10} = 0.46$$

Application of Bayes' Rule and the Theorem of Total Probability

- Suppose widgets are shipped to a store from three different factories F_1, F_2 and F_3 . Suppose the percentages of each shipment that are bad are 10%, 8%, and 3%, respectively.
- Suppose further that the percentages of the widgets supplied from these factories are 20%, 30%, and 50%, respectively.
- Now, suppose a customer selects a widget at random and finds that it is bad.
- What is the probability the widget came from factory F_2?



Solution

Let E_i denote the event the widget comes from factory $F_i, i = 1,2,3$. We are given

$$P(E_1) = 0.2,\ P(E_2) = 0.3,\ P(E_3) = 0.5$$

Let B denote the event that a widget selected at random is bad. We are also given the conditional probabilities

$$P(B|E_1) = 0.1, \qquad P(B|E_2) = 0.08, \qquad P(B|E_3) = 0.03$$

We are to determine $P(E_2|B)$. Using the Total Probability Theorem

$$P(B) = P(B|E_1)P(E_1) + P(B|E_2)P(E_2) + P(B|E_3)P(E_3)$$
$$(0.1)(0.2) + (0.08)(0.3) + (0.03)(0.5) = 0.059$$

Applying Bayes Rule we obtain

$$P(E_2|B) = \frac{P(B|E_2)P(E_2)}{P(B)} = \frac{(0.08)(0.3)}{0.059} = 0.406779661$$

Discrete Random Variables

A **discrete random variable** X is mapping from the sample space S to the real numbers.

For example, consider the sample space S where the outcomes correspond to rolling two dice. Then a natural random variable is the function R mapping an outcome to the sum of it values, i.e.,

$$R(i,j) = i + j,\ \ i,j \in \{1,2,3,4,5,6\}$$

Distribution of a Random Variable

The random variable *X* induces another sample space consisting of the range of values that the random variable takes. For example for the random variable *R* mapping a roll of two dice onto their sum, the induced sample space is

$$\{2,3,\ldots,12\}$$

The probability density of the induced sample space associated with X is called the **distribution** of X.

PSN. Give the distribution for the random variable *R* mapping a roll of two dice onto their sum.

How to make a small fortune from gambling?

Start with a large fortune.

Random Variables, Expectation and the Binomial Distribution

Textbook Reading:

Chapter 8, pp. 520-527

Discrete Random Variables

- Recall that a **discrete random variable** X is mapping from the sample space S to the real numbers.

- For example, consider the sample space S where the outcomes correspond to rolling two dice.

- Then a natural random variable is the function R mapping an outcome to the sum of it values, i.e.,

$$R(i,j) = i + j,\ \ i,j \in \{1,2,3,4,5,6\}$$

Distribution of a Random Variable

The random variable X induces another sample space consisting of the range of values that the random variable takes.

The probability density of the induced sample space associated with X is called the **distribution** of X.

For example for the random variable R mapping a roll of two dice onto their sum, the induced sample space is

$$\{2,3, \ldots, 12\}$$

The distribution of R is

$i+j$	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
$P(R = i + j)$	$\frac{1}{36}$	$\frac{2}{36}$	$\frac{3}{36}$	$\frac{4}{36}$	$\frac{5}{36}$	$\frac{6}{36}$	$\frac{5}{36}$	$\frac{4}{36}$	$\frac{3}{36}$	$\frac{2}{36}$	$\frac{1}{36}$

Binomial Distribution

Consider a sample space corresponding to an n-trial Bernoulli process, i.e., the outcomes correspond to n trials. where we associate the probability p of success on each trial and $q = 1 - p$ of failure

The trials can be represented by flipping a coin where the “success” is a head with probability p and “failure” is a tail with probability $q = 1 - p$.

Probability of an outcome

The probability of an outcome s

$$p(s) = p^k q^{n-k}$$

where k is the number of successes. This is the probability density function for the Binomial Distribution.

Example of Sample Space for $n = 4$

$$S = \{HHHH, HHHT, HHTH, HTHH, THHH, HHTT,$$
$$HTHT, HTTH, THHT, THTH, TTHH, HTTT$$
$$THTT, TTHT, TTTH, TTTT\}$$

PSN. Verify that $p(s) = p^k q^{n-k}$ is a probability density function for S where $p = q = \frac{1}{2}$, i.e., show that the probabilities over all the outcomes in S sum to 1.

Probability Density Function

- We now show in general that $p(s) = p^k q^{n-k}$ determines a probability density function.
- To verify this we must show that the sum of the probabilities over all the outcomes in $s \in S$ is 1.

Proof.

Using the binomial theorem

$$\sum_{s \in S} p(s) = \sum_{k=0}^{n} C(n,k) p^k q^{n-k} = (p+q)^n = 1^n = 1$$

Binomial Distribution

Let X map an outcome onto the number of successes. Then the distribution of X is called the **Binomial Distribution**.

For example, for 4 trials flipping a coin

$X(HHHH) = 4, X(HTTH) = 2,\ X(TTTT) = 0$

It follows from the definition of X that

$$P(X = k) = b(k; n, p) = C(n, k)p^k q^{n-k}$$

Expectation

One of the most important concepts in probability theory is that of the expectation $E(X)$ of a random variable X on a (finite) sample space S. The mean or **expectation** or **expected value** of X is defined by

$$E(X) = \sum_{s \in S} X(s) p(s)$$

Example – Rolling a die

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

$$R(i) = i, \qquad i = 1,2,\ldots,6$$

$$E(R) = \frac{1}{6}(1+2+3+4+5+6) = \frac{21}{6} = \frac{7}{2}.$$

Example– Rolling two dice

Compute expectation of R the sum of two dice

$$S = \{(1,1),(1,2),\ldots,(6,6)\}$$

$$R(i,j) = i + j, \qquad i,j = 1,2,\ldots,6$$

$E(R) =$

$$\frac{1}{36}(2+3+4+5+6+7) + \frac{1}{36}(3+4+5+6+7+8)$$

$$+\frac{1}{36}(4+5+6+7+8+9) \ +\frac{1}{36}(5+6+7+8+9+10)$$

$$+\frac{1}{36}(6+7+8+9+10+11) \ +\frac{1}{36}(7+8+9+10+11+12)$$

$$= 7$$

Computing Expectation using Distribution of Random Variable

An alternative way to compute expectation, which is often easier is using the distribution of the random variable

$$E(X) = \sum_x xP(X = x)$$

PSN. Use this formula to compute the expectation of the sum of two dice and compare to the result we derived on the previous slide.

Expectation of Binomial Distribution

Theorem. Suppose X is the binomial random variable corresponding to a given n and p. Then

$$E(X) = np$$

Proof

$$E(X) = \sum_{k=0}^{n} kP(X = k)$$

$$\sum_{k=0}^{n} kC(n,k)p^k q^{n-k} = np$$

We now verify that latter equality using the binomial theorem and calculus.

By the Binomial Theorem

$$(px+q)^n = \sum_{k=0}^{n} C(n,k)(px)^k q^{n-k}$$

Taking the derivative of both sides we obtain

$$np(px+q)^{n-1} = \sum_{k=0}^{n} kC(n,k)p^k x^{k-1} q^{n-k}$$

Setting $x = 1$ and using the fact that $p + q = 1$ we have

$$np = np(p1+q)^{n-1} = \sum_{k=0}^{n} kC(n,k)p^k q^{n-k}$$

Total Expectation Theorem

Let $F_1, F_2, \ldots, F_n$ be events that partition S and let X be a random variable on S. Then

$$E(X) = E(X|F_1)P(F_1) + E(X|F_2)P(F_2) + \cdots + E(X|F_n)P(F_n)$$

Expectation of sum of random variables

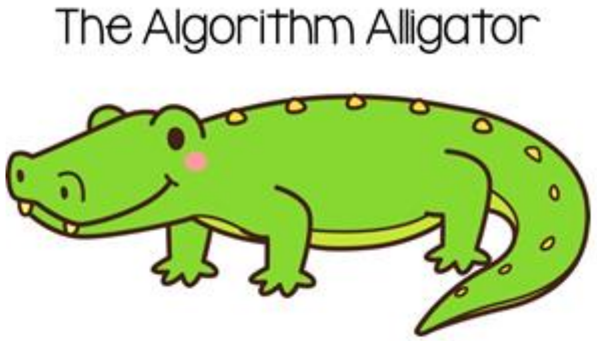
Let $X_1, X_2, \ldots, X_n$ be any random variables on S. Then

$$E(X_1 + X_2 + \cdots + X_n) = E(X_1) + E(X_2) + \cdots + E(X_n)$$

I have a special trick for getting the result I want in a coin flip.

It's not perfect, but it works about 50% of the time.

Application of Probability Theory to Algorithms



Definition of an Algorithm

An *algorithm* is a finite, nonambiguous sequence of steps for solving a problem.

Note. A formal definition of an algorithm requires the concept of a Turing Machine.

Two main applications of Probability Theory in the Design and Analysis of Algorithms

1) Probabilistic analysis of the computing time of an algorithm

2) Probabilistic design of algorithms

Average Computing Time of an Algorithm

- Define the random variable τ maps the sample space S_n of all inputs *I* of size *n* onto the computing time of the algorithm for that input. The expected computing time for an input of size *n* denoted *A*(*n*) is given by

$$A(n) = E(\tau).$$

- $A(n)$ is often referred to as the **average computing time** or **average complexity**.

- Often the computing time is measured by counting a **basic operation**, so that τ(*I*) is the number of times the algorithm performs the basic operation for input *I*.

Formula Average Complexity

Let p_i denote the number of basic operations algorithm performs for an input of size n, i.e., $p_i = P(\tau = i)$. Then

$$A(n) = \sum_{i=B(n)}^{W(n)} ip_i$$

where $B(n)$ and $W(n)$ denote the fewest and most basic operations performed by the algorithm for an input size n. They are called the best-case and worst-case complexities.

Example – Linear Search

function *LinearSearch* ($L[0{:}n-1], X$)

Input: $L[0{:}n-1]$ (a list of size n), X (a search item)

Output: returns index of first occurrence of X in the list, or -1 if X is not in the list

```
    for $i \leftarrow 0$ to $n-1$ do
        if $X = L[i]$ then
            return($i$)
        endif
    endfor
    return(-1)
end LinearSearch
```

Analysis of Computing Time

- The basic operation of *LinearSearch* is comparison
- The fewest comparisons performed by *LinearSearch* is one comparison when the input *X* is the first element in the list, i.e., $B(n) = 1$
- The most it performs is n comparison when *X* is not in the list or occurs only in the last position, i.e., $W(n) = n$.
- To obtain the average computing time we use the formula

$$A(n) = \sum_{i=B(n)}^{W(n)} ip_i = \sum_{i=1}^{n} ip_i$$

Average Complexity (Computing Time) of *LinearSearch*

- To simplify the discussion of the average behavior of *LinearSearch*, we assume that the search element X is in the list $L[0: n - 1]$ and is equally likely to be found in any of the *n* positions.
- Note that i comparisons are performed when X is found at position i in the list. Thus, the probability that *LinearSearch* performs i comparisons is given by $p_i = 1/n$.
- Substituting these probabilities in the formula from previous slide we obtain

$$A(n) = \sum_{i=1}^{n} ip_i = \sum_{i=1}^{n} i\frac{1}{n} = \frac{n(n+1)}{2} \times \frac{1}{n} = \frac{n+1}{2}$$

PSN

Suppose that the search element *X* is twice as likely to be found in the first half of the list as the second half. Assume (1) the size *n* of the list is even, (2) *X* occurs in the list exactly once, (3) there is an equal probability of *X* being found in any position in the first half and, (4) there is an equal probability of *X* being found in any position in the second half.

Average Complexity of *LinearSearch* when search element *X* may not occur in list *L*

- Let p = probability that X is in list L
- Then the probability X is not in L is $1 – p$.
- Applying the Total Expectation Theorem

$$\begin{aligned} A(n) &= \mathrm{E}(\tau) \\ &= \mathrm{E}(\tau \mid X \text{ in } L) \times P(X \text{ in } L) + \mathrm{E}(\tau \mid X \text{ not in } L) \times P(X \text{ not in } L) \\ &= \frac{n+1}{2} \times p + n(1 - p) \end{aligned}$$

The last step involves substituting the formula from the previous slide when X is in L and substituting n in the case when X is not in L.

Probabilistic Algorithms

- The algorithms that we have considered so far are all **deterministic**.
- They leave nothing to chance. Running a deterministic algorithm time after time with the same input will produce identical results each time.
- On the other hand, a **probabilistic** algorithm contains steps that make random choices by invoking a random (or pseudorandom) number generator.
- Thus, they are subject to the laws of chance. In particular, a probabilistic algorithm can perform differently for two runs with the same input.

Four Main Categories of Probabilistic Algorithms

- Randomizations of deterministic algorithms
- Monte Carlo algorithms
- Las Vegas algorithms
- Numerical probabilistic algorithms

Description of Types of Probabilistic Algorithms

1. **Randomization of a deterministic algorithm** results by replacing certain steps that made canonical choices by steps that make these choices in some random fashion. Randomization is done to break the connection between a particular input and worst-case behavior, and thereby homogenize the expected behavior of inputs to the algorithm.
2. **Monte Carlo algorithms** often produce solutions very quickly, but only guarantee correctness with high probability.
3. **A Las Vegas algorithm** never outputs an incorrect solution but has some probability of reporting a failure to produce a solution.
4. **Numerical probabilistic algorithms** were among the first examples of introducing randomness into the design of algorithms. A classic example is the estimation of π obtained by throwing darts at a square and recording how many darts land inside a circle inscribed in the square.

Advantage of Probabilistic Algorithms

- In practice, obtaining solutions with high probability is almost as satisfactory as the foolproof guarantee provided by a deterministic algorithm.
- For many important problems, such as prime testing, the most efficient algorithms currently known for their solutions are probabilistic.

Expected Number of Basic Operations

- Because running the algorithm twice with the same input I may result in a different number of basic operations being performed, $\tau(I)$ is no longer well defined.
- Instead, what is relevant is the expected number, $\tau_{\text{exp}}(I)$, of basic operations performed by the algorithm for input I with respect to the random choices made by the algorithm.
- As with any expectation, if we run the algorithm many times with fixed input I, then we can expect that the algorithm performs $\tau_{\text{exp}}(I)$ basic operations on average. If the algorithm performs many random choices, then even for a single run we can expect that the number of basic operations performed is very close to $\tau_{\text{exp}}(I)$.

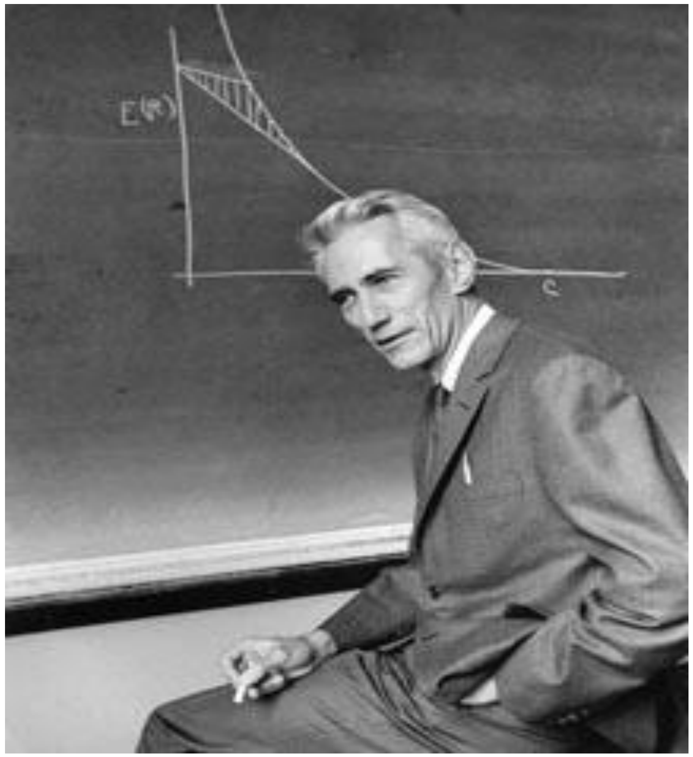
Alternate Definition of Algorithm

Shannon Entropy

“*Information is the resolution of uncertainty*” Shannon

Claude E. Shannon, 1916-2001

1937 Masters thesis, EE Dept, MIT
A symbolic analysis of relay and switching circuits
Introduced application of Boolean algebra to logic circuits, and vice versa.
Very influential in digital circuit design.
"Most important Masters thesis of the century"



1940 PhD, Math Dept, MIT
An algebra for theoretical genetics
To analyze the dynamics of Mendelian populations.

Joined Bell Labs in 1940.

"A mathematical theory of cryptography" 1945/1949
"A mathematical theory of communication" 1948

Seminal Publications of Shannon

In 1948, Claude E. Shannon published the paper A Mathematical Theory of Communication, which is seen as the foundation of modern information theory.

In 1949, Shannon published Communication Theory of Secrecy Systems which relates cryptography to information theory, and should be seen as the foundation of modern cryptography.

Both papers derive from a technical report, A Mathematical Theory of Cryptography, written by Shannon in 1945. In this report, Shannon defined, and mathematically proved, perfect secrecy.

Measuring Information

Shannon's definition of the information obtained on being told the outcome x_i of a probabilistic experiment X:

$$I(X = x_i) = \log_2(\frac{1}{p_X(x_i)})$$

where $p_X(x_i)$ is the probability of the event $X = x_i$.

The unit of measurement (when the log is base-2) is the **bit** (**b**inary **i**nformation uni**t** --- not the same as binary digit!).

1 bit of information corresponds to $p_X(x_i) = 0.5$. So, for example, when the outcome of a *fair* coin toss is revealed to us, we have received 1 bit of information.

Examples

We're drawing cards at random from a standard 52-card deck. Elementary outcome: card that's drawn, probability 1/52, information $\log_2(52/1) = 5.7$ bits. For an event comprising M such (mutually exclusive) outcomes, the probability is $M/52$.

Q. If I tell you the card is a spade ♠, how many bits of information have you received?

A. Out of 52 equally probable cards, $M = 13$ are spades ♠, so probability of drawing a spade is 13/52, and the amount of information received is $\log_2(52/13) = 2$ bits.

This makes sense, we can encode one of the 4 (equally probable) suits using 2 binary digits, e.g.,

00 = ♠ , 01 = ♥ , 10 = ♦ , 11 = ♣ .

Q. If instead I tell you the card is a seven, how much info?

A. $N = 52$, $M = 4$, so info = $\log_2(52/4) = \log_2(13) = 3.7$ bits

Properties of Information definition

- A lower−probability outcome yields higher information
- A highly informative outcome does not necessarily mean a more valuable outcome, only a more surprising outcome, i.e., there's no intrinsic value being assessed (can think of information as degree of surprise).
- Often used fact: The information in independent events is additive.

Expected Information as Uncertainty or Entropy

Consider a discrete random variable X, which may represent the set of possible symbols to be transmitted at a particular time, taking possible values $x_1, x_2, \ldots, x_N$, with respective probabilities $p_X(x_1), p_X(x_2), \ldots, p_X(x_N)$.

The *entropy* $H(X)$ of X is the expected (or mean or average) value of the information obtained by learning the outcome of X:

$$H(X) = \sum p_X(x_i) I(X = x_i) = \sum_{i=1}^{N} p_X(x_i) \log_2\left(\frac{1}{p_X(x_i)}\right)$$

When all the $p_X(x_i)$ are *equal* (with value $1/N$), then

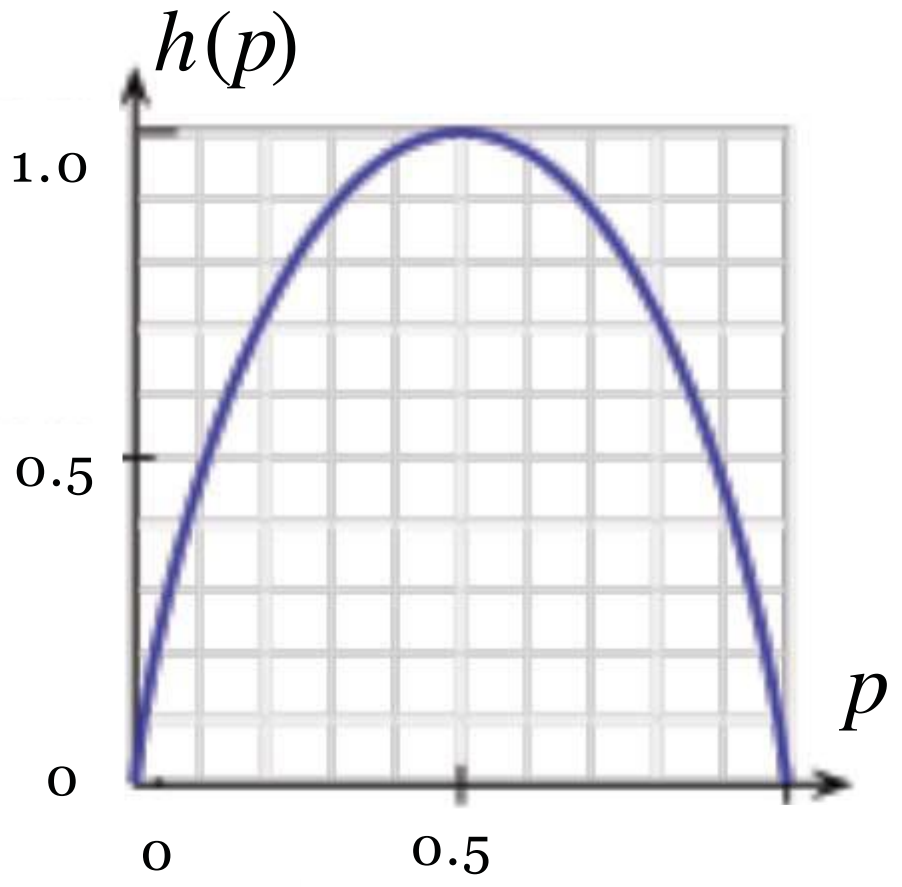
$$H(X) = \log_2 N \qquad \text{or} \qquad N = 2^{H(X)}$$

This is the maximum attainable value!

e.g., Binary entropy function $h(p)$

Heads (or C = 1) with probability *p*

Tails (or C = 0) with probability $1 - p$



$$H(C) = -p\log_2 p - (1-p)\log_2(1-p) = h(p)$$

Connection to (Binary) Coding

Suppose *p*=1/1024, i.e., very small probability of getting a Head, typically one Head in 1024 trials. Then

$$h(p) = (\frac{1}{1024}) \log_2 \left(\frac{1024}{1}\right) + \left(\frac{1023}{1024}\right) \log_2 \left(\frac{1024}{1023}\right) = .0012$$

.0012 bits of uncertainty or information per trial on average

So using 1024 binary digits (C = 0 or 1) to code the results of 1024 tosses of this particular coin seems inordinately wasteful, i.e., 1 binary digit per trial.

Q. Can we get closer to an average of .0112 binary digits/trial?
A. Yes!

Binary coding: Mapping source symbols to **binary digits.**

Significance of Entropy

Entropy (in bits) tells us the average amount of information (in bits) that must be delivered in order to resolve the uncertainty about the outcome of a trial. This is a lower bound on the number of binary digits that must, on the average, be used to encode our messages.

If we send fewer binary digits on average, the receiver will have some uncertainty about the outcome described by the message.

If we send more binary digits on average, we're wasting the capacity of the communications channel by sending binary digits we don't have to.

Achieving the entropy lower bound is the "gold standard" for an encoding (at least from the viewpoint of information compression).

Fixed-length Encodings

An obvious choice for encoding equally probable outcomes is to choose a fixed-length code that has enough sequences to encode the necessary information

- 96 printing characters → 7-”bit” ASCII
- Unicode characters → UTF-16
- 10 decimal digits → 4-”bit” BCD (binary coded decimal)

Fixed-length codes have some advantages:

- They are “random access” in the sense that to decode the *n*th message symbol one can decode the *n*th fixed-length sequence without decoding sequence 1 through $n - 1$.
- Table lookup suffices for encoding and decoding

Now consider:

$choice_i$	p_i	$log_2(1/p_i)$
"A"	1/3	1.58 bits
"B"	1/2	1 bit
"C"	1/12	3.58 bits
"D"	1/12	3.58 bits

The expected information content in a choice is given by the entropy:
= (.333)(1.58) + (.5)(1) + (2)(.083)(3.58) = 1.626 bits

Can we find an encoding where transmitting 1000 choices requires 1626 binary digits on the average?

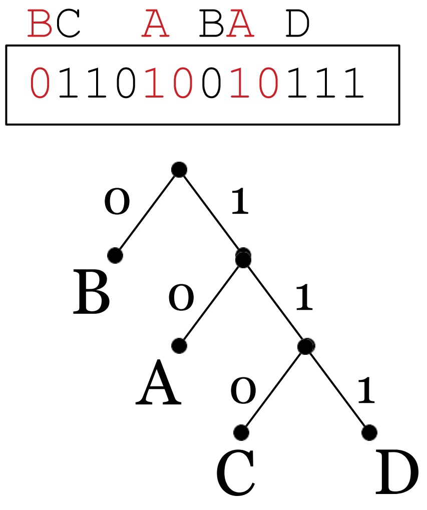
The "natural" fixed-length encoding uses two binary digits for each choice, so transmitting the results of 1000 choices requires 2000 binary digits.

Variable-length encodings

(David Huffman, in term paper for MIT graduate class, 1951)

Use shorter bit sequences for high probability choices, longer sequences for less probable choices

choice$_i$	p_i	*encoding*
"A"	1/3	10
"B"	1/2	0
"C"	1/12	110
"D"	1/12	111



Huffman Decoding Tree

Expected length
=(.333)(2)+(.5)(1)+(2)(.083)(3)
= 1.666 bits

Transmitting 1000 choices takes an average of 1666 bits… better but not optimal

Note: The symbols are at the leaves of the tree; necessary and sufficient for instantaneously decodability.

Why is it that programmers always confuse Halloween with Christmas?

Because 31 OCT = 25 DEC