Discrete Mathematics

Review

Shengfa Wang
sfwang@dlut.edu.cn

Course Schedule

Topics	Courses	note
Logic	**Propositional Logic**	****
	Predicate Logic	****
Set Theory	**Sets**	****
	Functions	***
	Relations	*****
Graph Theory	**Graphs**	***
	Trees	***
Review	All	*****

Propositional logic (命题逻辑)

Key Points

- Negation (否定) (NOT, ¬)
- Conjunction (合取) (AND, $\wedge$)
- Disjunction (析取) (OR, $\vee$)
- Conditional statement (条件) (Implication) (IF THEN→)

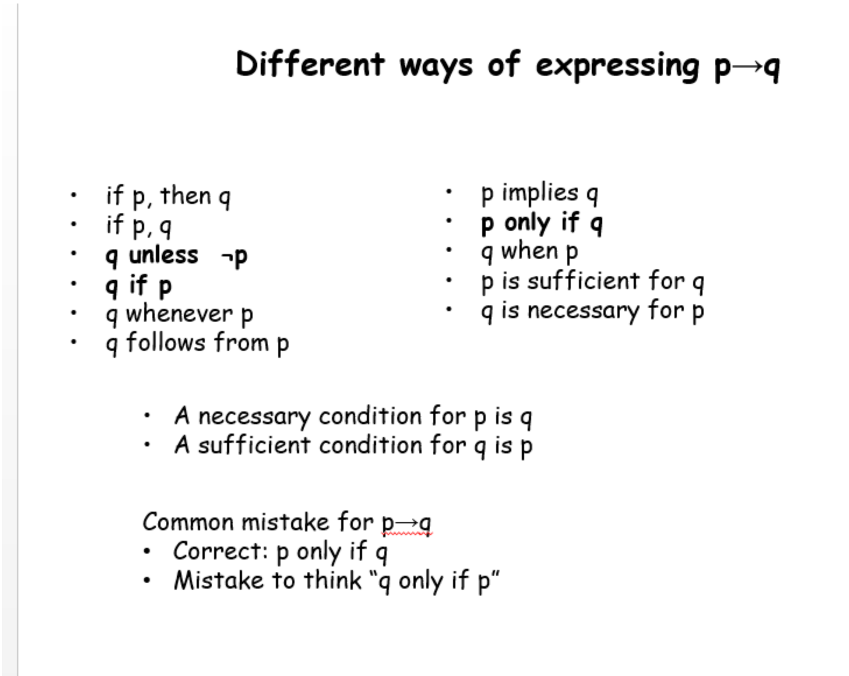
- Logical equivalence (逻辑等价)
  - Prove logical equivalence
- Normal forms (范式)*
  - Disjunctive normal form (析取范式)
  - Conjunctive normal form (合取范式)

Translating English to Logical Expressions

Problem: Translate the following sentence into propositional logic:

- "You can access the Internet from campus only if you are a computer science major or you are not a freshman."
- One solution:
  - p: "You can access the internet from campus,"
  - q: "You are a computer science major,"
  - r: "You are a freshman."

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



Different ways of expressing p→q

- if p, then q
- if p, q
- **q unless ¬p**
- **q if p**
- q whenever p
- q follows from p

- p implies q
- **p only if q**
- q when p
- p is sufficient for q
- q is necessary for p

- A necessary condition for p is q
- A sufficient condition for q is p

Common mistake for p→q
- Correct: p only if q
- Mistake to think "q only if p"

Equivalence proofs

Example: Show that $\neg(p \vee (\neg p \wedge q))$

is logically equivalent to $\neg p \wedge \neg q$

Solution:

$$
\begin{array}{llll}
\neg(p \vee (\neg p \wedge q)) & \equiv & \neg p \wedge \neg(\neg p \wedge q) & \text{by the second De Morgan law} \\
 & \equiv & \neg p \wedge [\neg(\neg p) \vee \neg q] & \text{by the first De Morgan law} \\
 & \equiv & \neg p \wedge (p \vee \neg q) & \text{by the double negation law} \\
 & \equiv & (\neg p \wedge p) \vee (\neg p \wedge \neg q) & \text{by the second distributive law} \\
 & \equiv & F \vee (\neg p \wedge \neg q) & \text{because } \neg p \wedge p \equiv F \\
 & \equiv & (\neg p \wedge \neg q) \vee F & \text{by the commutative law} \\
 & & & \text{for disjunction} \\
 & \equiv & (\neg p \wedge \neg q) & \text{by the identity law for } \mathbf{F}
\end{array}
$$

Simplifying statement

We can use logical rules to simplify a logical formula.

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

$$\equiv (\neg\neg p \vee \neg q) \wedge (p \vee q)$$ De Morgan

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

$$\equiv p \vee (\neg q \wedge q)$$ Distributive law

$$\equiv p \vee \text{False}$$

$$\equiv p$$

The De Morgan's Law allows us to always "**move the NOT inside**".

Principal disjunctive normal form (PDNF):

- **求主析取范式的方法：**
  - 先化成与其等价的析取范式；
  - 若析取范式的基本积中同一命题变元出现多次，则将其化成只出现一次；
  - 去掉析取范式中所有为永假式的基本积，即去掉基本积中含有形如 $p \wedge \neg p$的子公式的那些基本积；
  - 若析取范式中缺少某一命题变元如p，则可用公式$(p \vee \neg p) \wedge q \Leftrightarrow q$将命题变元P补充进去，并利用分配律展开，然后合并相同的基本积

Principal conjunctive normal form (PCNF)

- **求主合取范式的方法：**
  - 先化成与其等价的合取范式；
  - 若合取范式的基本积中同一命题变元出现多次，则将其化成只出现一次；
  - 去掉合取范式中所有为永真式的基本积，即去掉基本和中含有形如 $p \vee \neg p$的子公式的那些基本和；
  - 若合取范式中缺少某一命题变元如p，则可用公式$(p \wedge \neg p) \vee q \Leftrightarrow q$将命题变元P补充进去，并利用分配律展开，然后合并相同的基本和

Minterm vs Maxterm

- The relations between m_i and M_i are

$$M_i \Leftrightarrow \neg m_i \qquad m_i \Leftrightarrow \neg M_i$$

p,q,r	maxterms	$p \wedge q \vee r$		p,q,r	minterms	$p \wedge q \vee r$	
0,0,0	$p \vee q \vee r$	0	M_0	0,0,0	$\neg p \wedge \neg q \wedge \neg r$	0	m_0
0,0,1	$p \vee q \vee \neg r$	1	M_1	0,0,1	$\neg p \wedge \neg q \wedge r$	1	m_1
0,1,0	$p \vee \neg q \vee r$	0	M_2	0,1,0	$\neg p \wedge q \wedge \neg r$	0	m_2
0,1,1	$p \vee \neg q \vee \neg r$	1	M_3	0,1,1	$\neg p \wedge q \wedge r$	1	m_3
1,0,0	$\neg p \vee q \vee r$	0	M_4	1,0,0	$p \wedge \neg q \wedge \neg r$	0	m_4
1,0,1	$\neg p \vee q \vee \neg r$	1	M_5	1,0,1	$p \wedge \neg q \wedge r$	1	m_5
1,1,0	$\neg p \vee \neg q \vee r$	1	M_6	1,1,0	$p \wedge q \wedge \neg r$	1	m_6
1,1,1	$\neg p \vee \neg q \vee \neg r$	1	M_7	1,1,1	$p \wedge q \wedge r$	1	m_7

Translate CNF to DNF

Let CNF of A be

$$(P \vee Q \vee \neg R) \wedge (P \vee \neg Q \vee \neg R)$$

Find the DNF of A.

Solution:
The CNF of A is $M_1 \wedge M_{3,}$ so the DNF can be written as

$$\sum (0, 2, 4, 5, 6, 7)$$

And thus we have

$$(\neg P \wedge \neg Q \wedge \neg R) \vee (\neg P \wedge Q \wedge \neg R) \vee (P \wedge \neg Q \wedge \neg R)$$
$$\vee (P \wedge \neg Q \wedge R) \vee (P \wedge Q \wedge \neg R) \vee (P \wedge Q \wedge R)$$

Find the normal forms of $(p\to\neg q)\to\neg r$

Solution:

$(p\to\neg q)\to\neg r$

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

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

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

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

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

$\Leftrightarrow\prod(1,3,5)$

/*其中∏表示求合取*/

$\Leftrightarrow\sum(0,2,4,6,7)$

/*即该公式是可满足的，应存在与其等价的主析取范式*/

Predicate Logic (谓词逻辑)

Key Points

- Predicates (谓词)
- Quantifiers (量词)*
  - Universal quantifier (全称量词)
  - Existential quantifier (存在量词)
- Nested quantifiers (嵌套量词)*

Logically Equivalent

Show that $\neg\forall x(P(x) \rightarrow Q(x))$ and $\exists x(P(x) \land \neg Q(x))$ are logically equivalent.

Solution:

- By De Morgan’s law for universal quantifiers, we know that $\neg\forall x(P(x) \rightarrow Q(x))$ and $\exists x(\neg(P(x) \rightarrow Q(x)))$ are logically equivalent.
- It is also known that $\neg(P(x) \rightarrow Q(x))$ and $P(x) \land \neg Q(x)$ are logically equivalent for every x.
- Because we can substitute one logically equivalent expression for another in a logical equivalence, it follows that $\neg\forall x(P(x) \rightarrow Q(x))$ and $\exists x(P(x) \land \neg Q(x))$ are logically equivalent.

Nested quantifiers

- Nested quantifiers are often necessary to express the meaning of sentences in English as well as important concepts in computer science and mathematics.

  Example: "Every real number has an inverse" is

  $$\forall x\ \exists y(x + y = 0)$$

  where the domains of x and y are the real numbers.
- We can also think of nested propositional functions:

  $\forall x\ \exists y(x + y = 0)$ can be viewed as $\forall x\ Q(x)$ where $Q(x)$ is $\exists y\ P(x, y)$ where $P(x, y)$ is $(x + y = 0)$

Translating nested quantifiers into English

Example 1: Translate the statement

$$\forall x\ (C(x) \lor \exists y\ (C(y) \land F(x, y)))$$

where $C(x)$ is "x has a computer," and $F(x,y)$ is "x and y are friends," and the domain for both x and y consists of all students in your school.

Solution: Every student in your school has a computer or has a friend who has a computer.

Example 2: Translate the statement

$$\exists x \forall y\ \forall z\ ((F(x, y) \land F(x,z) \land (y \neq z)) \rightarrow \neg F(y,z))$$

Solution: There is a student none of whose friends are also friends with each other.

Translating English into predicate logic

Example : Translate "The sum of two positive integers is always positive" into a logical expression.

Solution:

1. Rewrite the statement to make the implied quantifiers and domains explicit:

   "For every two integers, if these integers are both positive, then the sum of these integers is positive."

2. Introduce the variables x and y, and specify the domain, to obtain:

   "For all positive integers x and y, x + y is positive."

3. The result is:

   $$\forall x \; \forall y \; ((x > 0) \land (y > 0) \rightarrow (x + y > 0))$$

   where the domain of both variables consists of all integers

量词演算规则

- 量词对∧、∨的分配律

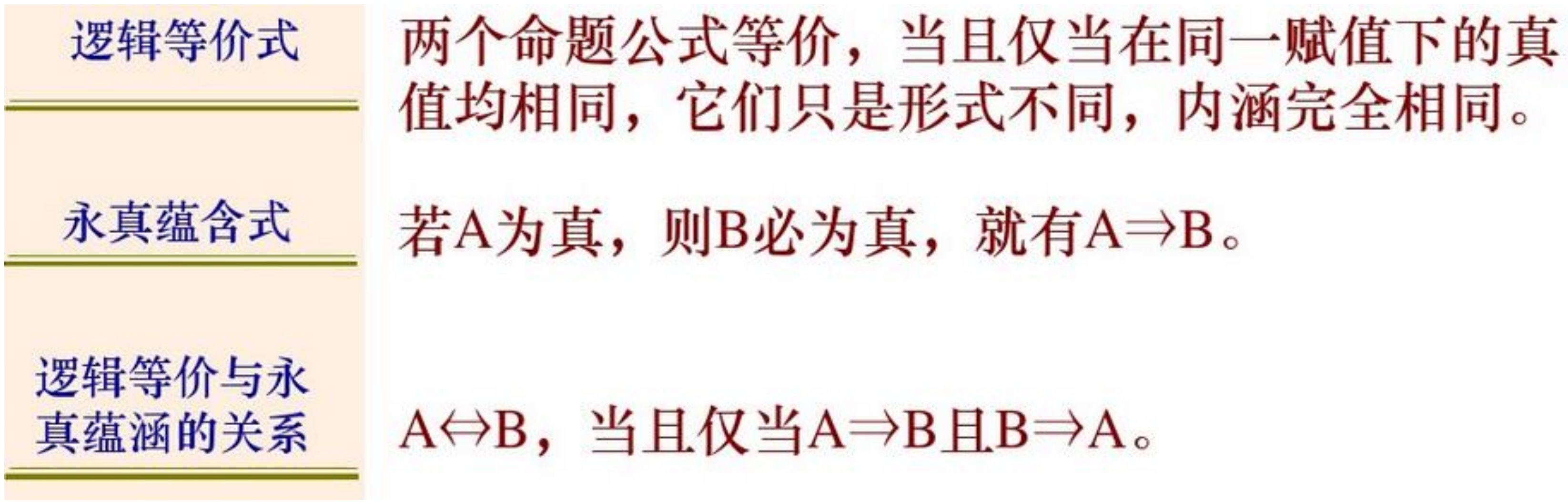
  $\forall x(A(x) \land B(x)) \Leftrightarrow \forall xA(x) \land \forall xB(x)$,

  $\exists x(A(x) \lor B(x)) \Leftrightarrow \exists xA(x) \lor \exists xB(x)$

  $\forall xA(x) \lor \forall xB(x) \Rightarrow \forall x(A(x) \lor B(x))$,

  $\exists x(A(x) \land B(x)) \Rightarrow \exists xA(x) \land \exists xB(x)$

  ($\Rightarrow$永真蕴含式)



逻辑等价式	两个命题公式等价，当且仅当在同一赋值下的真值均相同，它们只是形式不同，内涵完全相同。
永真蕴含式	若A为真，则B必为真，就有A⇒B。
逻辑等价与永真蕴涵的关系	A⇔B，当且仅当A⇒B且B⇒A。

量词演算规则

- 多个量词的使用
  - 各量词按照从左到右的顺序读出，不得随意颠倒顺序，但是当出现的量词相同时，量词的顺序可以颠倒

Example:

A(x,y): x read y. The domains of x and y are all human and books, respectively

$\forall x \forall y A(x,y)$

$\forall y \forall x A(x,y)$

$\exists x \exists y A(x,y)$

$\exists y \exists x A(x,y)$

$\forall y \exists x A(x,y)$: T

$\exists x \forall y A(x,y)$: F

$\exists x \forall y A(x,y) \Rightarrow \forall y \exists x A(x,y)$

记忆规律

$$\begin{tikzcd}
 & \boxed{(\forall x)(\forall y)} \arrow[r, leftrightarrow, "B_1"'] \arrow[dl, "B_2"] & \boxed{(\forall y)(\forall x)} \arrow[dr, "B_3"] & \\
\boxed{(\exists y)(\forall x)} \arrow[d, "B_4"] & & & \boxed{(\exists x)(\forall y)} \arrow[d, "B_5"] \\
\boxed{(\forall x)(\exists y)} \arrow[dr, "B_6"] & & & \boxed{(\forall y)(\exists x)} \arrow[dl, "B_7"'] \\
 & \boxed{(\exists y)(\exists x)} \arrow[r, leftrightarrow, "B_8"] & \boxed{(\exists x)(\exists y)} &
\end{tikzcd}$$

Set

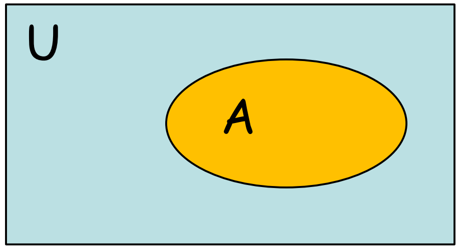
Key Points

- set, subset (proper subset 真子集)
- Cardinality (基数：Size of a Set)
- Set Operations
  - Union (并)
  - Intersection (交)
  - Difference (差)
  - Complement (补)
  - Symmetric Difference (对称差) (Option)

Sets

- Set = a collection of distinct **unordered** objects
- Members of a set are called **elements**
- How to determine a set
  - **Listing**
    - Example: $A = \{1,3,5,7\} = \{7, 5, 3, 1, 3\}$
  - **Description**
    - Example: $B = \{x \mid x=2k+1, 0 < k < 30\}$
  - **Venn Diagrams**

- A Venn diagram provides a graphic view of sets
- Venn diagrams are useful in representing sets and set operations which can be easily and visually identified.
- Various sets are represented by circles inside a big rectangle representing the universal set.

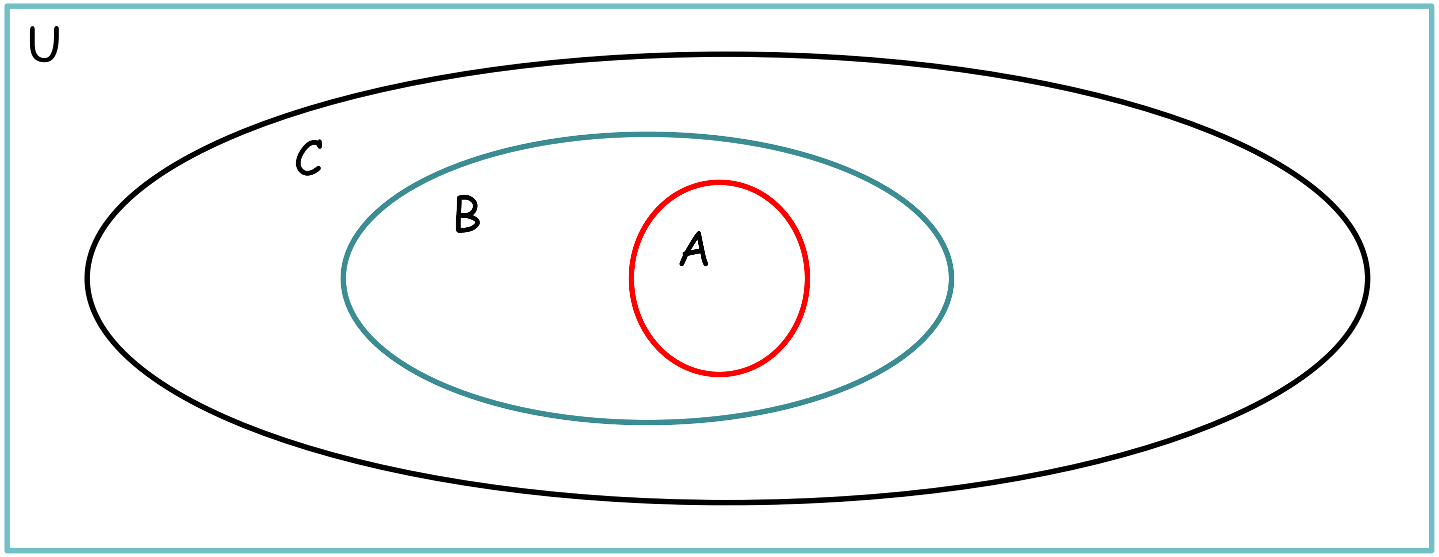


Subsets

- Useful rules:
- $A = B \Leftrightarrow (A \subseteq B) \wedge (B \subseteq A)$
- $(A \subseteq B) \wedge (B \subseteq C) \Rightarrow A \subseteq C$ (next Venn Diagram)

When "$\subset$" is used instead of "$\subseteq$", proper containment is meant. subset A of B is said to be a **proper subset** if A is not equal to B.
Notationally:

$$A \subset B \quad \Leftrightarrow \quad A \subseteq B \wedge \exists x \; (x \notin A \wedge x \in B)$$
$$\Leftrightarrow \quad \forall x \, (x \in A \rightarrow x \in B) \wedge \exists x \, (x \in B \wedge x \notin A)$$



Cardinality

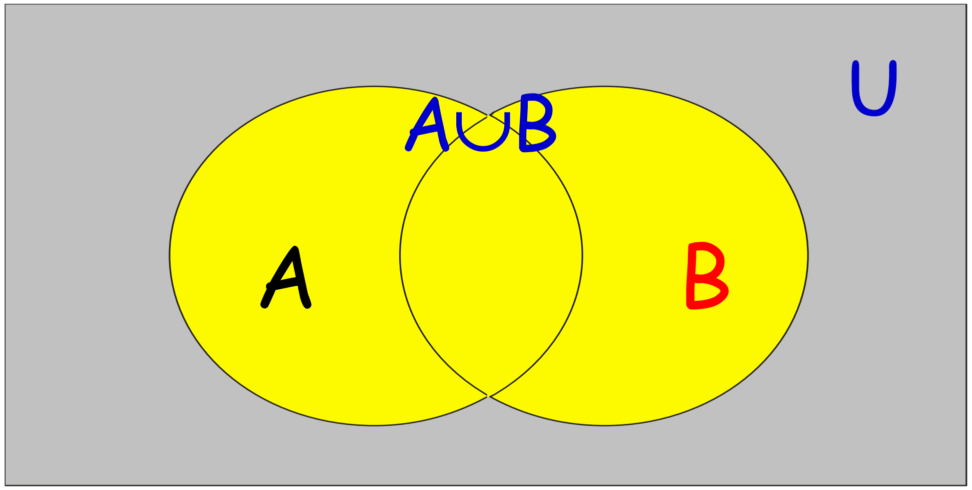
Hint: After eliminating the redundancies just look at the number of top level commas and add 1 (except for the empty set).

A:

1. $|\{1, -13, 4, -13, 1\}| = |\{1, -13, 4\}| = 3$
2. $|\{3, \{1,2,3,4\}, \emptyset\}| = 3$. To see this, set $S = \{1,2,3,4\}$. Compute the cardinality of $\{3,S, \emptyset\}$
3. $|\{\}| = |\emptyset| = 0$
4. $|\{ \{\}, \{\{\}\}, \{\{\{\}\}\} \}| = |\{ \emptyset, \{\}, \{\emptyset\}, \{\{\emptyset\}\}| = 3$

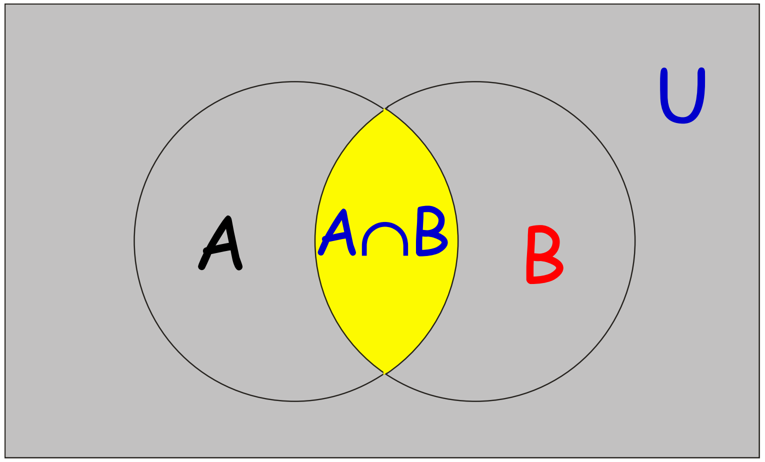
Set Operations

- Union: Elements in at least one of the two sets.
  - $A \cup B = \{x \mid x \in A \vee x \in B\}$
  - Example:
    - $A = \{a, b\}$, $B = \{b, c, d\}$
    - $A \cup B = \{a, b, c, d\}$



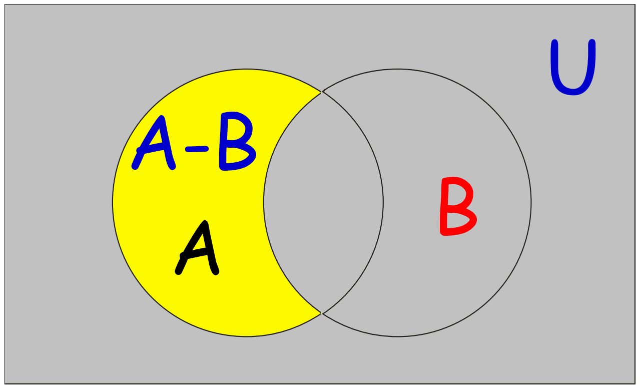
Set Operations

- Intersection: Elements in exactly one of the two sets.
  - $A \cap B = \{x \mid x \in A \wedge x \in B\}$
  - Example:
    - $A = \{a, b\}$, $B = \{b, c, d\}$
    - $A \cap B = \{b\}$



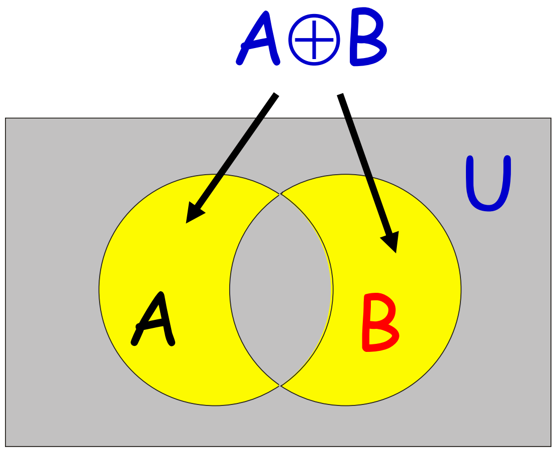
Set Operations

- Difference: Elements in first set but not second. Difference is also called the relative complement （相对补） of B in A.
  - $A-B = \{x \mid x \in A \wedge x \notin B\} = A \cap B^c$
  - Example
    - $A = \{a, b\}$, $B = \{b, c, d\}$
    - $A-B = \{a\}$



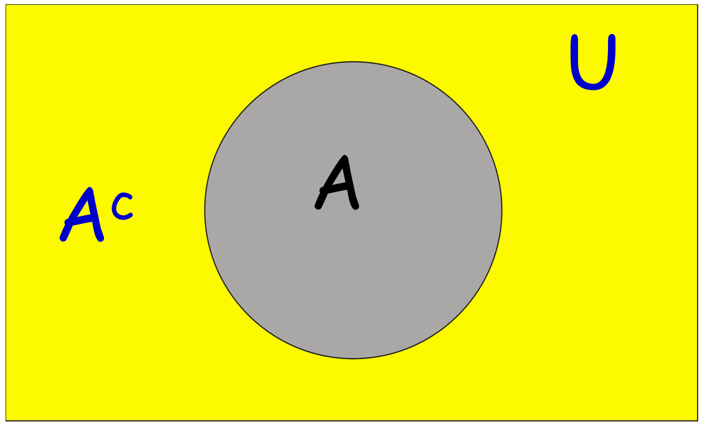
Set Operations

- Symmetric Difference: Elements in exactly one of the two sets.
  - $A \oplus B = \{ x \mid x \in A \oplus x \in B \} = (A-B) \cup (B-A) = (A \cup B)-(A \cap B)$
  - Example:
    - $A = \{a, b\}$, $B = \{b, c, d\}$
    - $A \oplus B = \{a,c,d\}$



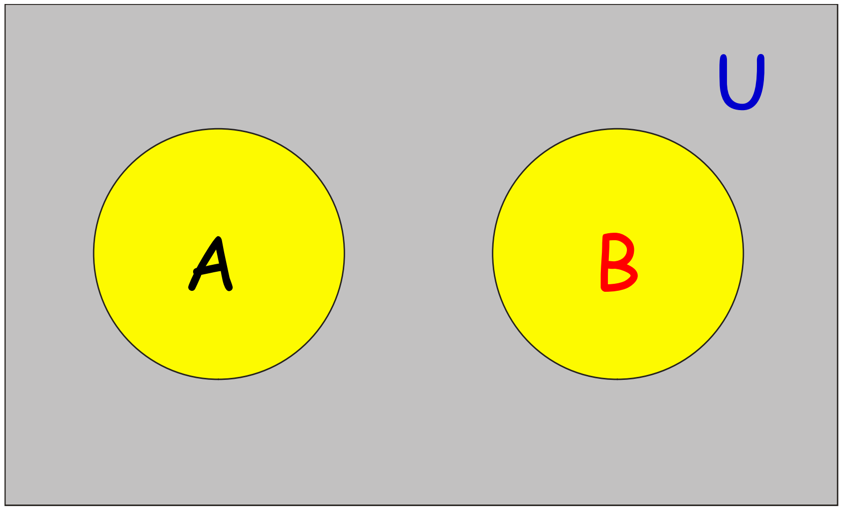
Set Operations

- Complement: Elements not in the set (unary operator).
  - $A^c = \{ x \mid x \notin A \}$
  - Example:
    - $U = N$, $A = \{250, 251, 252, \ldots\}$
    - $A^c = \{0, 1, 2, \ldots, 248, 249\}$



Disjoint Sets

- Disjoint: If *A* and B have no common elements, they are said to be disjoint.
  - $A \cap B = \varnothing$



Using Properties of Set Operations

- How can we prove $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$?
- Method I:
- $x \in A \cup (B \cap C)$
- $\Leftrightarrow x \in A \lor x \in (B \cap C)$
- $\Leftrightarrow x \in A \lor (x \in B \land x \in C)$
- $\Leftrightarrow (x \in A \lor x \in B) \land (x \in A \lor x \in C)$
- (distributive law for logical expressions)
- $\Leftrightarrow x \in (A \cup B) \land x \in (A \cup C)$
- $(A \cup B) \cap (A \cup C)$

Using Properties of Set Operations

- Method II: Membership table
  - 1 means "x is an element of this set"
  - 0 means "x is not an element of this set"

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

$S_1 \quad A \cap B \subseteq A$

$S_2 \quad A \cap B \subseteq B$

$S_3 \quad A \subseteq A \cup B$

$S_4 \quad B \subseteq A \cup B$

$S_5 \quad A - B \subseteq A$

$S_6 \quad A \oplus B \subseteq A \cup B$

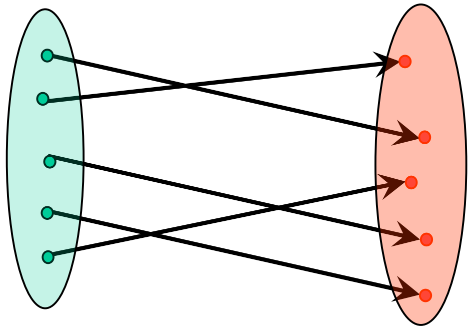
$$\left.\begin{aligned} &S_7 \quad A \cup B = B \cup A \\ &S_8 \quad A \cap B = B \cap A \\ &S_9 \quad A \oplus B = B \oplus A \end{aligned}\right\}\text{交换律}$$

$$\left.\begin{aligned} &S_{10} \quad A \cup (B \cup C) = (A \cup B) \cup C \\ &S_{11} \quad (A \cap B) \cap C = A \cap (B \cap C) \\ &S_{12} \quad (A \oplus B) \oplus C = A \oplus (B \oplus C) \end{aligned}\right\}\text{结合律}$$

Functions

More formally, we write $f : A \rightarrow B$

to represent that *f* is a function from set *A* to set *B*, which associates an element $f(a) \in B$ with an element $a \in A$.



The *domain (input)* of *f* is A.

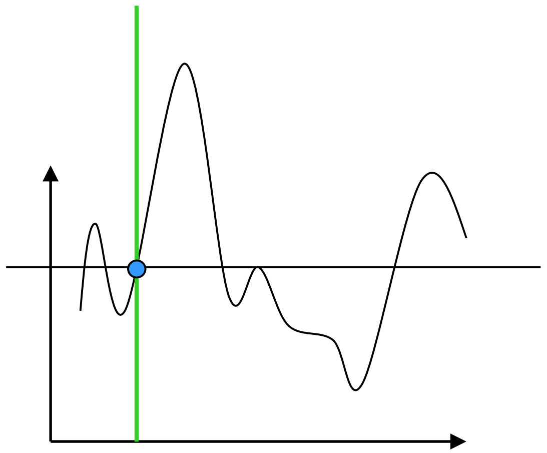
The *codomain (output)* of *f* is B.

Definition: For every input there is exactly one output.

Note: the input set can be the same as the output set, e.g. both are integers.

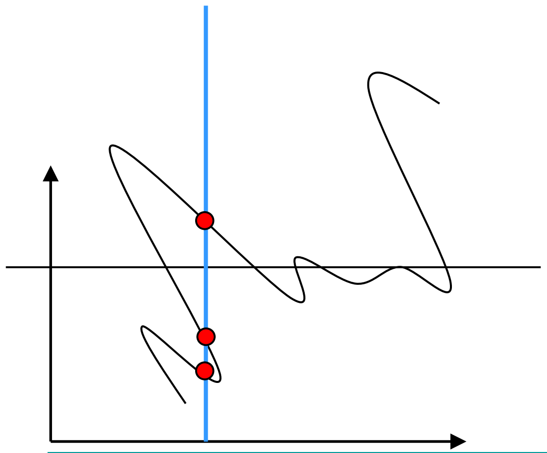
Algebraically speaking

- Note that such definitions on functions are consistent with what you have seen in your Calculus courses.



function

→ 1 intersection



not a function

→ violations when > 1

Key Points

- Properties of functions
- Composition of functions

Properties of Functions

- A function $f:A \to B$ is said to be one-to-one (or **injective**（单射）), if and only if $\forall x, y \in A\ (f(x) = f(y) \to x = y)$

- In other words: f is one-to-one if and only if it does not map two distinct elements of A onto the same element of B.

- How can we prove that a function f is one-to-one?
  - Whenever you want to prove something, first take a look at the relevant definition(s):

    $\forall x, y \in A\ (f(x) = f(y) \to x = y)$

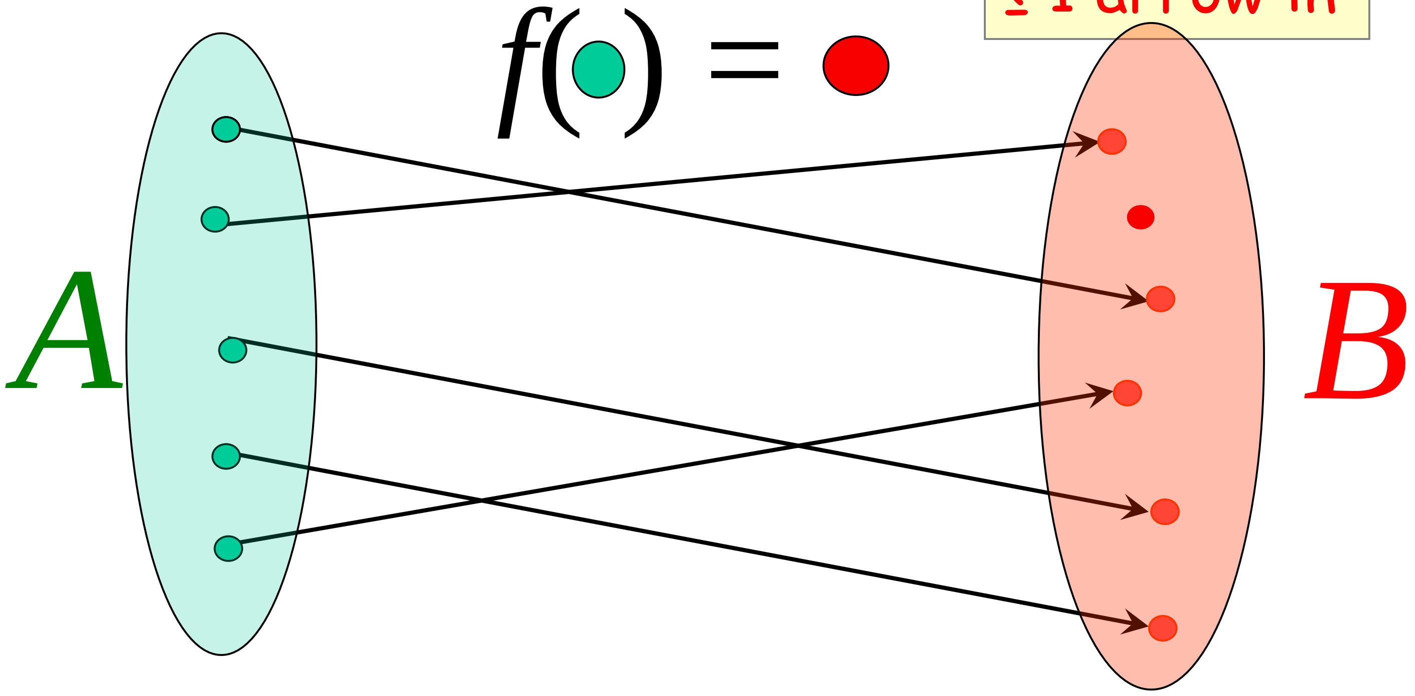
- Example:
  $f:\mathbf{R} \to \mathbf{R}$
  $f(x) = x^2$
- Disproof by counterexample:

$f(3) = f(-3)$, but $3 \neq -3$, so f is not one-to-one.

Injections

$f : A \to B$ is an *injection* if no two inputs have the same output.

≤ 1 arrow in

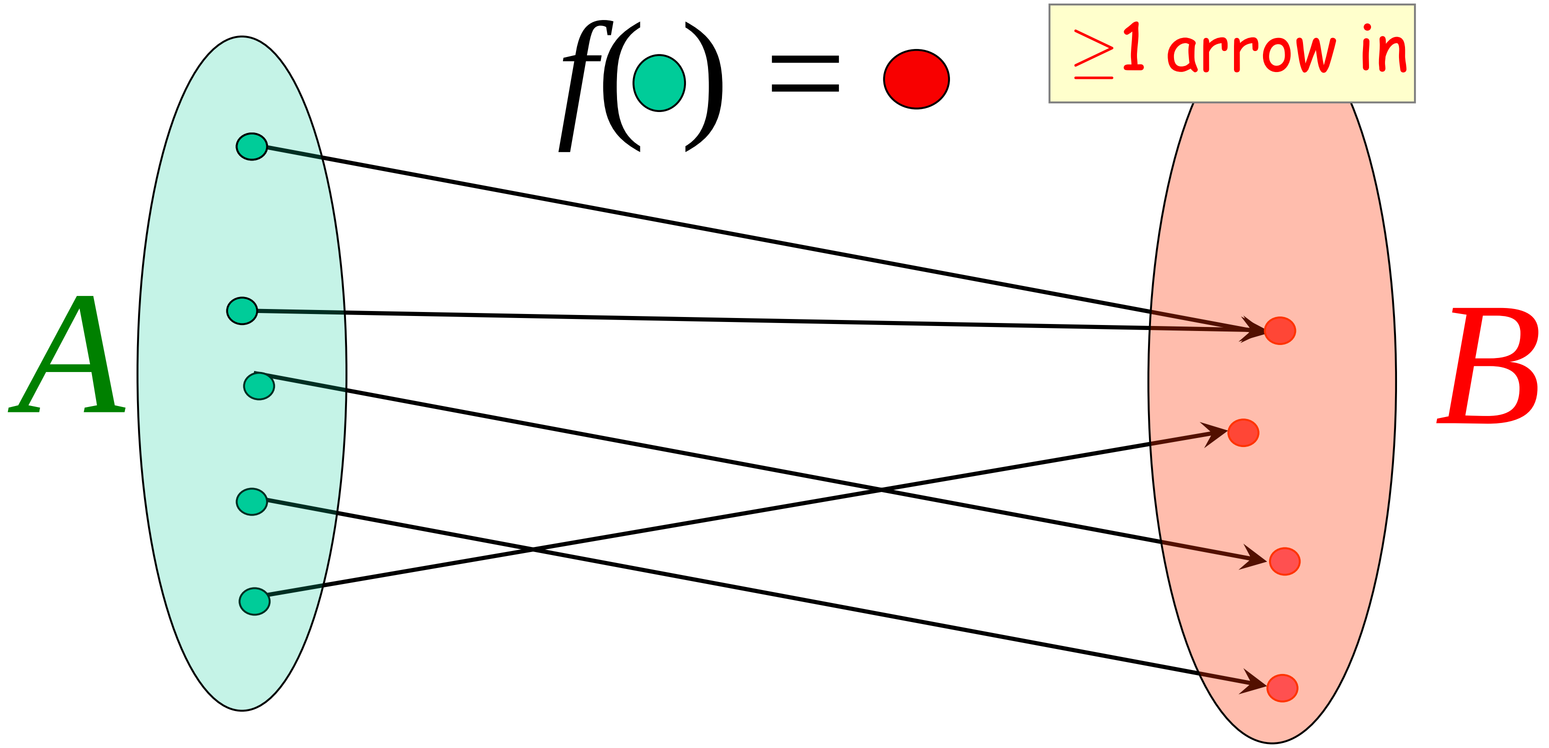


$$\boxed{|A| \leq |B|}$$

Surjections

$f : A \rightarrow B$ is a *surjection* if every output is possible.

$f(●) = ●$

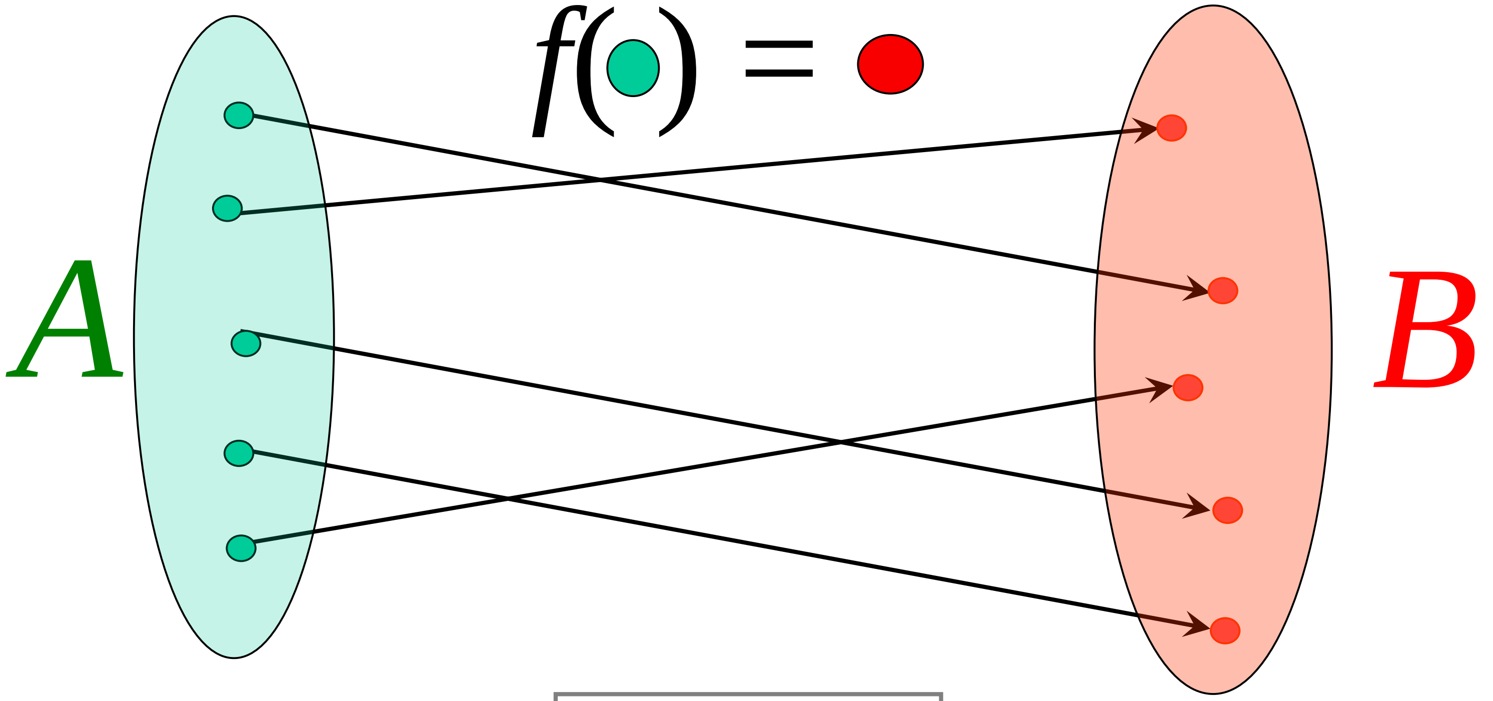


$$\boxed{|A| \geq |B|}$$

Bijections

$f : A \to B$ is a *bijection* if it is surjection and injection.

exactly one arrow in

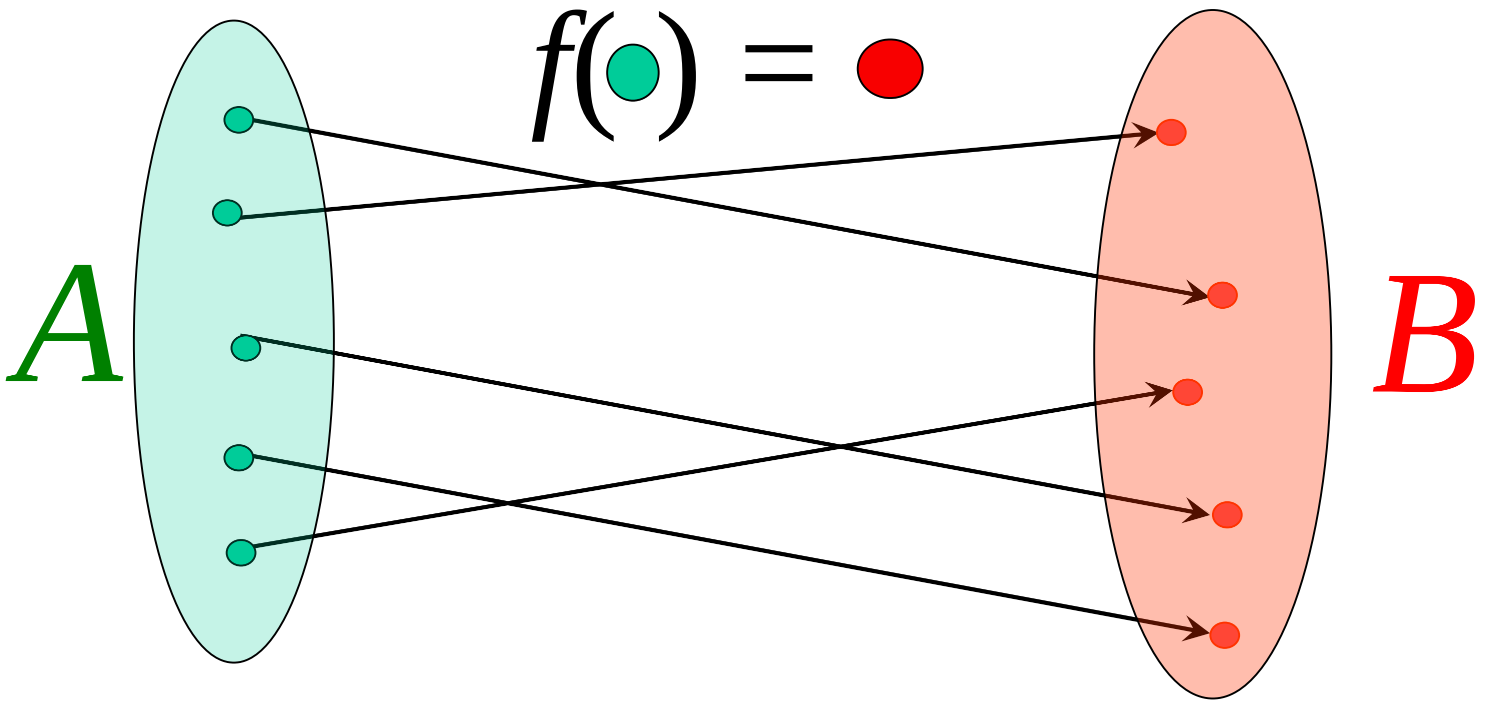


$\boxed{|A| = |B|}$

Inverse Function

Informally, an inverse function f^{-1} is to "undo" the operation of function f.

exactly one arrow in



There is an inverse function f^{-1} for f if and only if f is a bijection.

Composition

- The composition of two functions $g:A \to B$ and $f:B \to C$, denoted by $f \circ g$, is defined by $(f \circ g)(a) = f(g(a))$
- This means that
  - first, function g is applied to element $a \in A$, mapping it onto an element of B,
  - then, function f is applied to this element of B, mapping it onto an element of C.
  - Therefore, the composite function maps from A to C.

- Example:

  $f(x) = 7x - 4$, $g(x) = 3x$,

  $f:\mathbf{R} \to \mathbf{R}$, $g:\mathbf{R} \to \mathbf{R}$

  $(f \circ g)(5) = f(g(5)) = f(15) = 105 - 4 = 101$

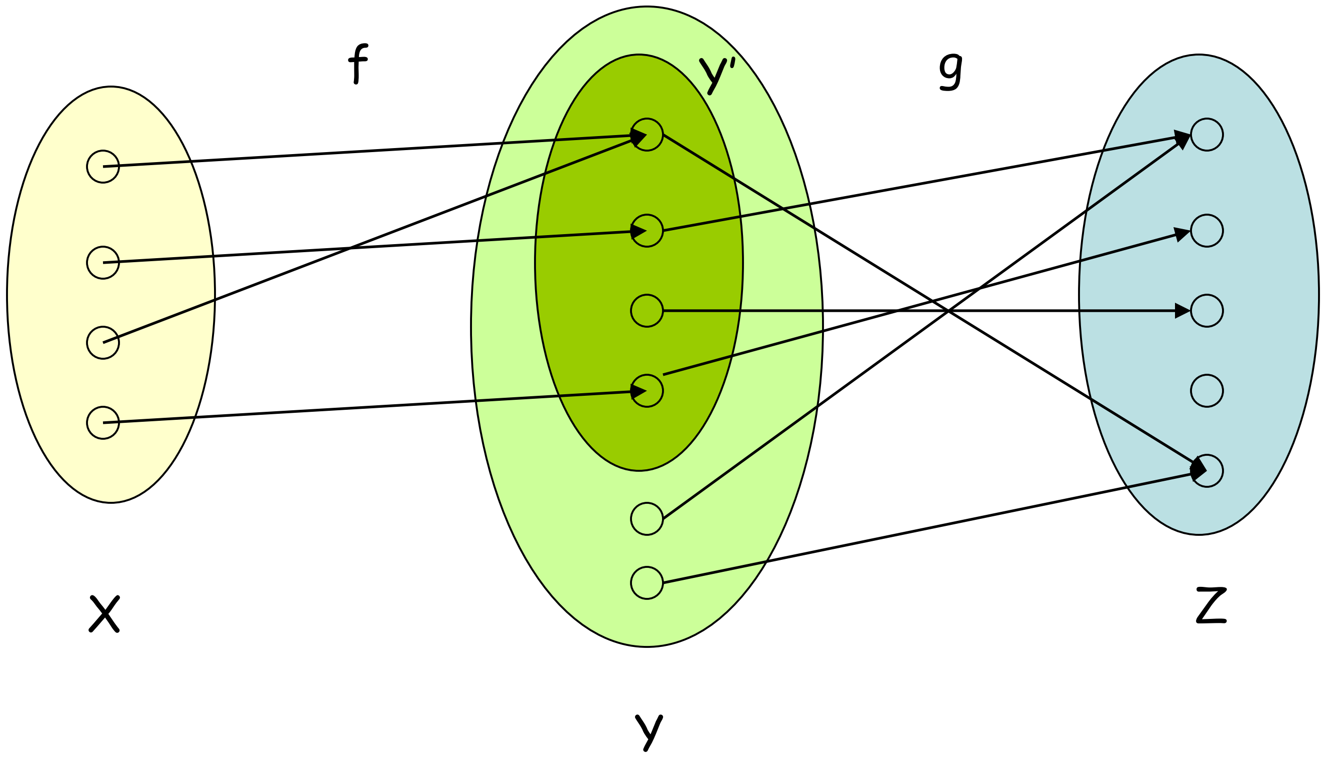
  $(f \circ g)(x) = f(g(x)) = f(3x) = 21x - 4$

Composition of Functions

Two functions f:X->Y', g:Y->Z so that Y' is a subset of Y,

then the composition of f and g is the function $g \circ f$: X->Z, where

$$g \circ f(x) = g(f(x)).$$



Summary

- f,g injective, f$\circ$g injective
- f,g surjective, f$\circ$g surjective
- f, g bijective, f$\circ$g bijective

Modulus operator (模运算)

- Let x be a nonnegative integer and y a positive integer
- r = x mod y is the remainder when x is divided by y
  - Examples:
    1 = 13 mod 3
    6 = 234 mod 19
    4 = 2002 mod 111
  - Basically, remove the complete y's and count what's left
  - mod is called the modulus operator

取模运算（“Modulo Operation”）和取余运算（“Complementation ”）两个概念有重叠的部分但又不完全一致。区别在于对负整数进行除法运算时操作不同。

如：
1.求 整数商： c = a/b;
2.计算模或者余数： r = a - c*b.

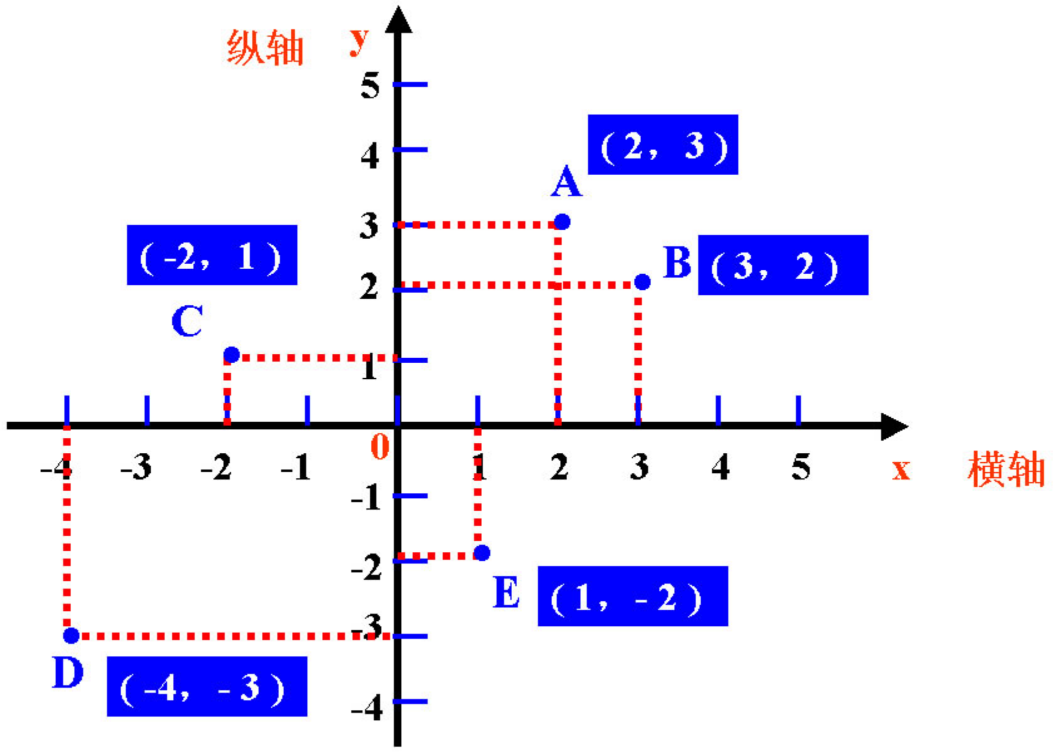
求模运算和求余运算在第一步不同: 取余运算在取c的值时，向0 方向舍入；而取模运算在计算c的值时，向负无穷方向舍入。
当a,b符号不一致时。求模结果的符号和b一致，求余运算结果的符号和a一致。

Key Points

- Cartesian Product, Relations and Binary Relations
- Properties of relations
  - reflexive (自反) , irreflexive (反自反), symmetric (对称), antisymmetric (反对称)，transitive (传递)
- Representing Binary Relations
- Operations of relations
- Closure (闭包)

Cartesian product (笛卡尔积)

- If $A_1, A_2, \ldots, A_m$ are nonempty sets, then the **Cartesian Product** of them is the set of all ordered m-tuples $(a_1, a_2, \ldots, a_m)$, where $a_i \in A_i$, $i = 1, 2, \ldots m$.
- Denoted $A_1 \times A_2 \times \ldots \times A_m = \{(a_1, a_2, \ldots, a_m) \mid a_i \in A_i, i = 1, 2, \ldots m\}$



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

$(2,3) \neq (3,2)$

Using matrices to denote Cartesian product

- If $A = \{1, 2, 3\}$ and $B = \{a, b, c\}$, find $A \times B$
- $A \times B = \{(1,a), (1,b), (1,c), (2,a), (2,b), (2,c), (3,a), (3,b), (3,c)\}$

- For Cartesian Product of two sets, you can use a matrix to find the sets.
- Example: Assume $A = \{1, 2, 3\}$ and $B = \{a, b, c\}$. The table below represents $A \times B$.

	a	b	c
1	(1, a)	(1, b)	(1, c)
2	(2, a)	(2, b)	(2, c)
3	(3, a)	(3, b)	(3, c)

The cardinality of the Cartesian Product equals the product of the cardinality of all of the sets:

$$| A_1 \times A_2 \times \dots \times A_m | = | A_1 | \cdot | A_2 | \cdot \dots \cdot | A_m |$$

Relations: Subsets of Cartesian products

A:

1. Database $\subseteq$ {Names}$\times${Foods}$\times${Colors}$\times${Jobs}

Definition: Let A_1, A_2, … , A_n be sets. An n-**ary relation** on these sets (in this order) is a subset of $A_1 \times A_2 \times \dots \times A_n$.

Most of the time we consider n = 2 in which case have a **binary relation** and also say the relation is "**from** A_1 to A_2".

With this terminology, all functions are relations, but not vice versa.

Functions as specific relations

- Recall that a function f from a set A to a set B assigns **exactly one** element of B to each element of A
- The graph of f is the set of ordered pairs (a, b) such that b=f(a)
- Because the graph of f is a subset of A x B, it is a relation from A to B
- A relation can be used to express one-to-many relationship between the elements of the sets A and B where an element of A may be related to more than one element of B
- A function represents a relation where exactly one element of B is related to each element of A
- Relations are a generalization of functions

Binary relations *

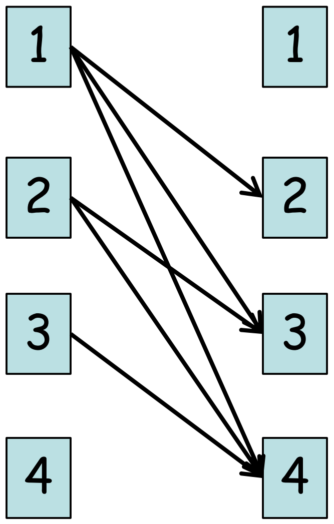
- Given two sets A and B, its Cartesian product $A \times B$ is the set of all ordered pairs (a, b) where $a \in A$ and $b \in B$
  - In symbols $A \times B = \{(a, b) | a \in A \text{ and } b \in B\}$
- **Definition**: Let A and B be sets. A **binary relation** R from a set A to a set B is a subset of the Cartesian product $A \times B$.
- In other words, for a binary relation R we have $R \subseteq A \times B$. We use the notation aRb to denote that $(a, b) \in R$ and $a\not{R}b$ to denote $(a, b) \notin R$.

Binary relations

- When (a, b) belongs to R, a is said to be related to b by R.
- **Example**: A = {1, 2, 3} and B = {a, b}
  - R = {(1, a), (1, b), (2, b), (3, a)} is a relation between A and B. 3 is related to a by R.
- **Example**: Let P be a set of people, C be a set of cars, and D be the relation describing which person drives which car(s).

Binary relations on a set

- Solution: R={(1,2),(1,3),(1,4),(2,3),(2,4),(3,4)}

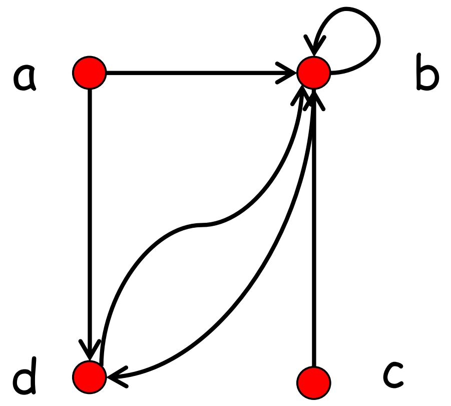


R	1	2	3	4
1		x	x	x
2			x	x
3				x
4				

Representing binary relations

- We have many ways of representing binary relations. We now take a closer look at two ways of representation: **Boolean (zero-one) matrices** and **directed graphs**.

	C1	C2	C3	C4	C5
P1	0	0	0	1	1
P2	1	0	0	1	0
P3	0	0	0	0	0



The matrix of a relation

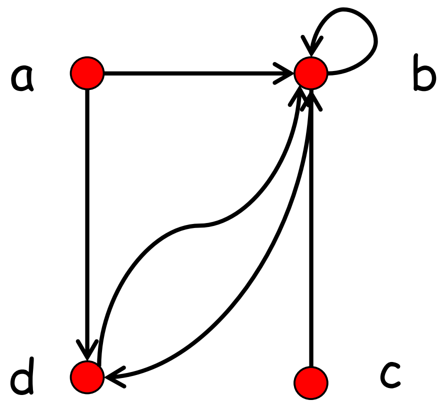
- If R is a relation from a set X to itself and M_R is the matrix of R, then M_R is a square matrix.
- Example: Let X = {2,3,4,5} and R = {(x,y) | x+y divides by 3}. Then :

$$M_R = \begin{array}{c|cccc} & 2 & 3 & 4 & 5 \\ \hline 2 & 0 & 0 & 1 & 0 \\ 3 & 0 & 1 & 0 & 0 \\ 4 & 1 & 0 & 0 & 1 \\ 5 & 0 & 0 & 1 & 0 \end{array}$$

fill it

Representing relations - Digraphs

- **Example**:
  - Display the digraph with V = {a, b, c, d}, E = {(a, b), (a, d), (b, b), (b, d), (c, a), (c, b), (d, b)}.



- An edge of the b called a loop form (b, b) is loop.

Properties of relations

Special properties for relation on a set A:

- **Reflexive** : every element is self-related. I.e. aRa for all a $\in A$
- **Symmetric** : order is irrelevant. I.e. for all a,b $\in A$ aRb iff bRa
- **Transitive** : when a is related to b and b is related to c, it follows that a is related to c. I.e. for all a,b,c $\in A$ aRb and bRc implies aRc

Q: Which of these properties hold for:

1) "Siblinghood" 2) "<" 3) "$\leq$"

Properties of relations - Warnings

- Warnings: there are additional concepts with confusing names
  - **Antisymmetric**: not equivalent to “not symmetric”. Meaning: it’s never the case for $a \neq b$ that both aRb and bRa hold.
  - **Asymmetric**: also not equivalent to “not symmetric”. Meaning: it’s never the case that both aRb and bRa hold.
  - **Irreflexive**: not equivalent to “not reflexive”. Meaning: it’s never the case that aRa holds.

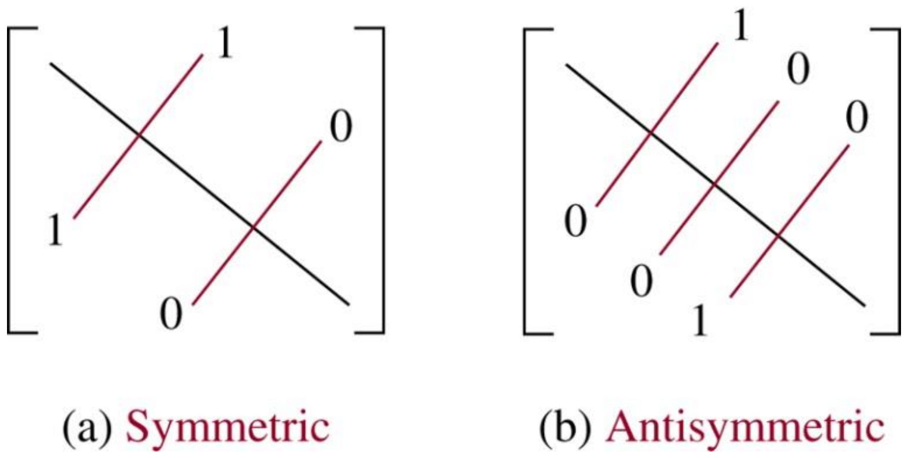
Reflexive

- The matrix of a relation on a set, which is a square matrix, can be used to determine whether the relation has certain properties
- Recall that a relation R on A is reflexive if $(a,a) \in R$. Thus R is reflexive if and only if $(a_i, a_i) \in R$ for $i=1,2,\ldots,n$
- Hence R is reflexive iff $m_{ii}=1$, for $i=1,2,\ldots, n$.
- R is reflexive if all the elements on the main diagonal of M_R are 1

$$\begin{bmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & \cdot & & & & \\ & & & & \cdot & & & \\ & & & & & \cdot & & \\ & & & & & & 1 & \\ & & & & & & & 1 \end{bmatrix}$$

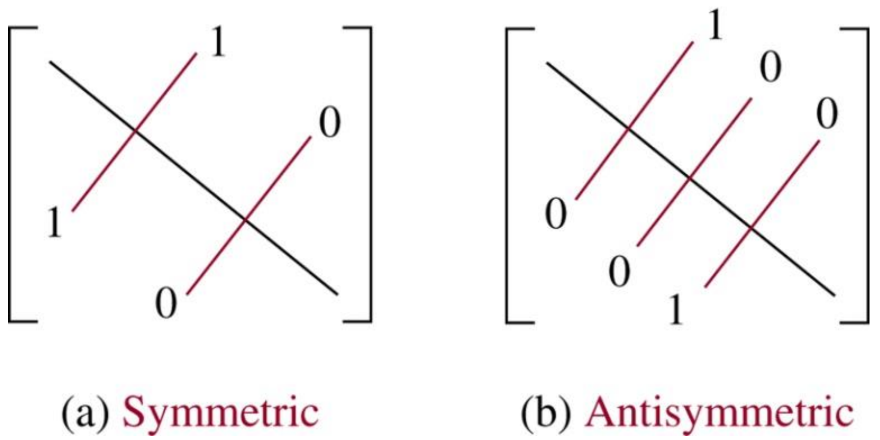
Symmetric

- The relation R is symmetric if (a,b)∈R implies that (b,a)∈R
- In terms of matrix, R is symmetric if and only m_{ji}=1 whenever m_{ij}=1, i.e., $M_R=(M_R)^T$
- R is symmetric iff M_R is a symmetric matrix

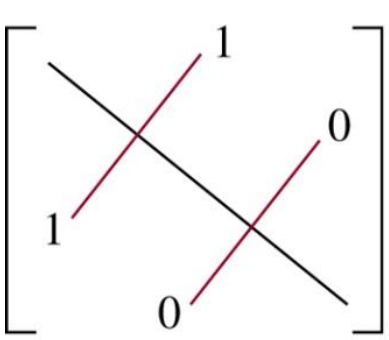


Antisymmetric

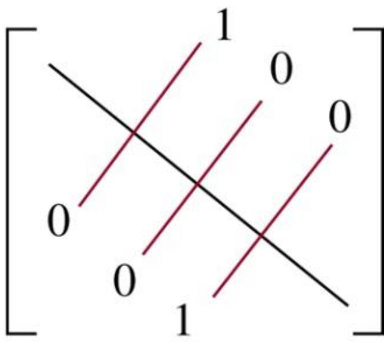
- The relation R is antisymmetric if (a,b)∈R and (b,a)∈R imply a=b
- The matrix of an antisymmetric relation has the property that if $m_{ij}=1$ with $i\neq j$, then $m_{ji}=0$
- In other words, either $m_{ij}=0$ or $m_{ji}=0$ when $i\neq j$



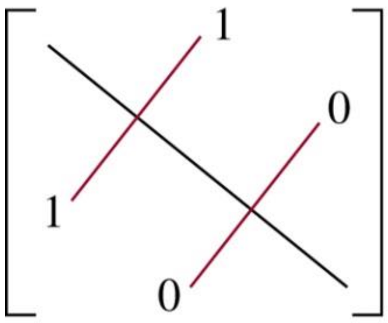
Visualization



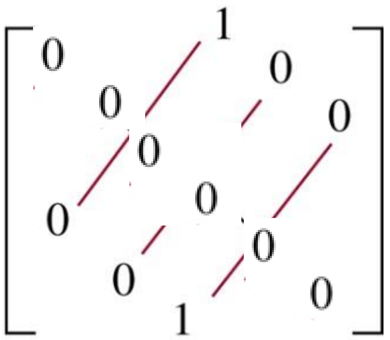
(a) Symmetric



(b) Antisymmetric



(a) Symmetric



(b) Asymmetric

Symmetric, Asymmetric and Antisymmetric

- 对称的(symmetric)：对所有的aRb,都有bRa
- 非对称的(not symmetric)：存在一些aRb,满足$b\cancel{R}a$
- 不对称的(asymmetric)：对所有的aRb,都有$b\cancel{R}a$
- 非不对称的(not asymmetric)：存在一些aRb,满足bRa
- 反对称的(antisymmetric)：对所有的aRb和bRa,都有a=b
- 非反对称的(not antisymmetric)：存在一些a≠b,满足aRb和bRa

可见：

- (1) asymmetric→not symmetric,而not symmetric不能得出asymmetric
- (2) asymmetric→antisymmetric,而antisymmetric不能得出asymmetric

举例1：A={1,2,3,4},R={(1,2),(2,2),(3,4),(4,1)},则：

- R是非对称的(not symmetric),因为(1,2)属于R,而(2,1)不属于R;
- R是非不对称的(not asymmetric),因为(2,2)属于R
- R是反对称的(antisymmetric),因为对于任意a≠b,不存在(a,b)和(b,a)都属于R

Useful summary

- Let R be a relation on a set *A*, i.e. R is a subset of the Cartesian product $A \times A$
  - R is **reflexive** if (x, x)∈R for every x∈A
  - R is **irreflexive** if (x, x)∉R for every element x∈A.
  - R is **symmetric** if for all x, y ∈A such that (x,y) ∈R then (y, x) ∈R
  - R is **antisymmetric** if for all x, y∈A such that x≠y, if (x, y) ∈R then (y, x) ∉R
  - R is **transitive** if (x, y) ∈R and (y, z)∈R imply (x, z)∈R

Operators on Relations

- Operators on Sets
- Inversion
- Composite *

Combining Relations

- Let R_1 be a relation from X to Y
- Let R_2 be a relation from Y to Z
- **Definition**: The composition of R_1 and R_2, denoted $R_2 \bullet R_1$ (or $R_2 \odot R_1$), is a relation from X to Z defined by $\{(x, z) \mid (x, y) \in R_1$ and $(y, z) \in R_2$ for some $y \in Y\}$
- In other words, if relation R_1 contains a pair (x, y) and relation R_2 contains a pair (y, z), then $R_2 \bullet R_1$ contains a pair (x, z).

$R_2 \bullet R_1$ contains a pair (x, z) : $x \; R_2 \bullet R_1 \; z$

Definition: If R is a relation from A to B, and S is a relation from B to C then the **composite** of R and S is the relation $S \bullet R$ (or just SR) from A to C defined by setting a $(S \bullet R)$c if and only if there is some b such that aRb and bSc.

Notation is weird because generalizing functional composition: $f \bullet g (x) = f (g (x))$.

Combining Relations

Composite relation: $S \circ R$

$$(a, b) \in S \circ R \leftrightarrow \exists x : (a, x) \in R \wedge (x, b) \in S$$

Note:

$$(a, b) \in R \wedge (b, c) \in S \rightarrow (a, c) \in S \circ R$$

Example:

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

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

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

Examples

- Let X and Y be relations on A = {1, 2, 3, …}.
- X = {(a, b) | b = a + 1} "b equals a plus 1"
  (X = {(b, c) | c = b + 1} "c equals b plus 1")
- Y = {(a, b) | b = 3a} "b equals 3 times a"
  (Y = {(b, c) | c = 3b} "c equals 3 times b")

Y maps an element a to the element 3a, and afterwards X maps 3a to 3a + 1), resulting in X • Y= {(a,b) | b = 3a + 1}
X • Y= {(a,c) | c = 3a + 1} = {(a,b) | b = 3a + 1}

Y • X= {(a, b) | b = 3a + 3}

Power of relations

Power of relation: R^n

$$R^0 = I_A \quad R^1 = R \quad R^{n+1} = R^n \circ R$$

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

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

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

$$R^4 = R^3 \circ R = R^3$$

Power of relations

Theorem: Suppose $|A|=n$, R is a relation on A. Then there exists s and t, such that $R^s = R^t$, $0 < s,t < 2^{n^2}$

Proof:
For any k, R^k is the subset of AxA, as $|AxA| = n^2$, $|P(AxA)|=2^{n^2}$. Then we can list all the relations as $R^0, R^1, \dots , R^{2^{n^2}}$. It is easy to check that there are $2^{n^2}+1$ relations. So there must exist $R^s = R^t$, $0 < s,t < 2^{n^2}$

有穷集合上关系的幂序列式一个周期变化的序列!

Examples

X={a,b,c}， R_1, R_2, R_3, R_4, Show the power of these relations

$$R_1 = \{(a,b),(a,c),(c,b)\}$$

$$R_2 = \{(a,b),(b,c),(c,a)\}$$

$$R_3 = \{(a,b),(b,c),(c,c)\}$$

$$R_4 = \{(a,b),(b,a),(c,c)\}$$

$$R_1^2 = \{(a,b)\}, R_1^3 = \varnothing, R_1^4 = \varnothing, \cdots$$

$$R_2^2 = \{(a,c),(b,a),(c,b)\},$$

$$R_2^3 = \{(a,a),(b,b),(c,c)\} = R_2^0$$

$$R_2^4 = R_2, R_2^5 = R_2^2$$

$$R_2^6 = R_2^3, \cdots$$

$$R_3^2 = \{(a,c),(b,c),(c,c)\} = R_3^3 = R_3^4 = R_3^5 \cdots$$

$$R_4^2 = \{(a,a),(b,b),(c,c)\} = R_4^0$$

$$R_4^3 = R_4, R_4^5 = R_4^3 \cdots$$

Summary

Theorem: Let R be the relation on a set A. Then we have

R is reflexive iff $I_A \subseteq R$

R is irreflexive iff $I_A \cap R = \emptyset$

R is symmetric iff $R = R^{-1}$

R is antisymmetric iff $R \cap R^{-1} \subseteq I_A$

R is transitive iff $R \bullet R \subseteq R$

Theorem: A relation R is transitive if an only if $R^n \subseteq R$ for all $n = 1,2,3, \ldots$

Closures of Relations

R: X→X

关系**S**满足

R的 可传递 闭包

r(R)

s

t(R)

(1) S 是自 可传递的

(2)R⊆ S

(3)对任何 可传递 关系**S′**

R ⊆ S′ ⇨ **S⊆S′**

Reflexive Closures

- **Example**: Find the reflexive closure of relation R = {(1, 1),(1, 2), (2, 1), (3, 2)} on the set A={1,2,3}.
- **Solution**:
  - We know that any reflexive relation on A must contain the elements (1,1), (2,2),and (3, 3).
  - By adding (2, 2) and (3, 3) to R, we obtain the reflexive relation S, which is given by S = {(1, 1), (1, 2), (2, 1), (2, 2), (3, 2), (3, 3)}.
  - S is reflexive, contains R, and is contained within every reflexive relation that contains R.
  - Therefore S is reflexive closure of R.

Symmetric Closures

- **Example**: Find the symmetric closure of the relation $R = \{(a, b) \mid a > b\}$ on the set of positive integers.
- **Solution**:
  - The symmetric closure of R is given by $R \cup R^{-1} = \{(a, b) \mid a > b\} \cup \{(b, a) \mid a > b\} = \{(a, b) \mid a \neq b\}$

Transitive Closures

- **Example:** Find the transitive closure of the relation R = {(1, 3), (1, 4), (2, 1), (3, 2)} on the set A = {1, 2, 3, 4}.
- **Solution**:
  - R would be transitive, if for all pairs (a, b) and (b, c) in R there were also a pair (a, c) in R.
  - If we add the missing pairs (1, 2), (2, 3), (2, 4), and (3, 1), will R be transitive?

- No, because the extended relation R contains (3, 1) and (1, 4), but does not contain (3, 4).
- By adding new elements to R, we also add new requirements for its transitivity. We need to look at paths in digraphs to solve this problem.
- Imagine that we have a relation R that represents all train connections in the US.

Transitive Closures

- Therefore, R* is the union of R^n across all positive integers n:

$$R^* = \bigcup_{n=1}^{\infty} R^n = R^1 \bigcup R^2 \bigcup R^3 \cdots$$

Theorem: For a relation R on a set *A* with n elements, the transitive closure R* is given by:

$$R^* = R \cup R^2 \cup R^3 \cup \ldots \cup R^n$$

For matrices representing relations we have:

$$M_{R^*} = M_R \vee M_R^{[2]} \vee M_R^{[3]} \vee \ldots \vee M_R^{[n]}$$

Equivalence Relations

- Equivalence relations are used to relate objects that are similar in some way.
- **Definition**: A relation on a set *A* is called an equivalence relation if it is **reflexive**, **symmetric**, and **transitive**.
- Two elements that are related by an equivalence relation R are called equivalent.

Equivalence Relations

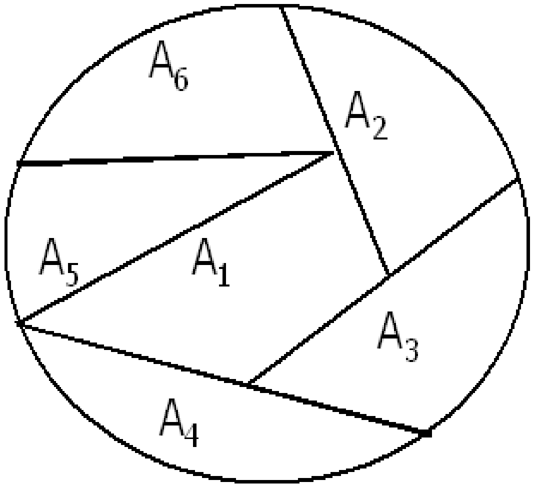
- **Example**: Suppose that R is the relation on the set of strings that consist of English letters such that aRb if and only if l(a)=l(b),where l(x) is the length of the string x. Is R an equivalence relation?
- **Solution**:
  - R is reflexive, because l(a) =l(a) and therefore aRa for any string a.
  - R is symmetric, because if l(a) = l(b) then l(b) = l(a), so if aRb then bRa.
  - R is transitive, because if l(a) = l(b) and l(b) = l(c), then l(a) = l(c), so aRb and bRc implies aRc.
- R is an equivalence relation.

Equivalence Classes

- **Definition**: Let R be an equivalence relation on a 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 denoted by $[a]_R$.
- When only one relation is under consideration, we will delete the subscript R and write [a] for this equivalence class.
- If $b \in [a]_R$, b is called a representative of this equivalence class.

Equivalence Classes

- **Theorem**: Let R be an equivalence relation on a set A. The following statements are equivalent:
  - 1. aRb
  - 2. [a] = [b]
  - 3. $[a] \cap [b] \neq \emptyset$
- **Definition**: A **partition** of a set S is a collection of disjoint nonempty subsets of S that have S as their union. In other words, the collection of subsets Ai, $i \in I$, forms a partition of S iff
  - 1. $A_i \neq \emptyset$ for $i \in I$
  - 2. $A_i \cap A_j = \emptyset$, if $i \neq j$
  - 3. $\cup_{i \in I} A_i = S$



划分的图形表示

Equivalence Classes

Examples: Let S be the set {u, m, b, r, o, c, k, s}. Do the following collections of sets partition S ?

- {{m, o, c, k}, {r, u, b, s}} yes.
- {{c, o, m, b}, {u, s}, {r}} no (k is missing).
- {{b, r, o, c, k}, {m, u, s, t}} no (t is not in S).
- {{u, m, b, r, o, c, k, s}} yes.
- {{b, o, o, k}, {r, u, m}, {c, s}} yes ({b,o,o,k} ={b,o,k}).
- {{u, m, b}, {r, o, c, k, s}, ∅} no (∅ not allowed).

Equivalence relations

- **Example**:
  - Consider set X = {1,2,…,13}. Define xRy as 5 divides x – y (i.e., x – y = 5k, for some int k). We can verify that R is reflexive, symmetric, and transitive. Here is how.
  - The equivalence class [1] consists of all x with xR1. Thus:
    - [1] = {x ∈ X | 5 divides x – 1} = {1, 6, 11}
  - Similarly:
    - [2] = {2, 7, 12}
    - [3] = {3, 8, 13}
    - [4] = {4, 9}
    - [5] = {5, 10}

Partial Order relations

- **Definition**: Let X be a set and R a relation on X, R is a partial order on X if R is **reflexive**, **antisymmetric** and **transitive**. A set X together with a partial ordering R is called a partially ordered set, or poset, or PO, and is denoted by (X, R).
- **Example**: Is $(x, y) \in R$ in partial order if x >= y?
  - Yes, since:
    - Reflexive: $(x, x) \in R$
    - Anti-symmetric: If $(x, y) \in R$ and $x \neq y$, then $(y, x) \notin R$
    - Transitive: If $(x, y) \in R$ and $(y, z) \in R$, then $(x, z) \in R$

Partial Order relations

- **Example**: Is the "inclusion relation" $\subseteq$ a partial ordering on the power set of a set S?
  - $\subseteq$ is reflexive, because $A \subseteq A$ for every set $A \in S$.
  - $\subseteq$ is antisymmetric, because if $A \neq B$, then $A \subseteq B \wedge B \subseteq A$ is false.
  - $\subseteq$ is transitive, because if $A \subseteq B$ and $B \subseteq C$, then $A \subseteq C$.
- Consequently, $(P(S), \subseteq)$ is a partially ordered set or poset.

Partial Order relations

- Let $x,y \in X$,
  - If (x,y) or (y,x) are in R, then x and y are **comparable**.
  - If $(x,y) \notin R$ and $(y,x) \notin R$, then x and y are **incomparable**.
  - **Definition**: If every pair of elements in X are comparable, then R is a **total order** on X.
    - In this case, X is called a totally ordered or linearly ordered set, and $\leq$ is called a total order or linear order. A totally ordered set is also called a **chain**.

Partial Order relations

- **Example**: Is (Z, ≤) a totally ordered poset?
  - Yes, because a ≤ b or b ≤ a for all integers a and b.
- **Example**: Is (Z^+, division) a totally ordered poset?
  - No, because it contains incomparable elements such as 5 and 7.

Partial Order relations

- In a poset the notation $a \preceq b$ denotes that $(a, b) \in R$.
- Note that the symbol $\preceq$ is used to denote the relation in any poset, not just the "less than or equal" relation.
- The notation $a \prec b$ denotes that $a \preceq b$, but $a \neq b$.
- If $a \prec b$ we say "a is less than b" or "b is greater than a".

Lexicographic Order

- How can we define a lexicographic ordering on the set of English words?
- This is a **special case** of an ordering of strings on a set constructed from a partial ordering on the set.
- We already have an ordering of letters (such as $a < b$, $b < c$, …), and from that we want to derive an ordering of strings.
- Let us take a look at the *general* case, that is, how the construction works in any poset.

Lexicographic Order

- **First step**: Construct a partial ordering on the Cartesian product of two posets,(A_1, $\leq_1$) and (A_2, $\leq_2$):
- $(a_1,a_2) < (b_1,b_2)$ if $(a_1 <_1 b_1) \vee [(a_1=b_1) \wedge (a_2 <_2 b_2)]$
- $(a_1,a_2) \leq (b_1, b_2)$ if $(a_1 <_1 b_1) \vee [(a_1 = b_1) \wedge (a_2 \leq_2 b_2)]$
- **Examples**:
  - In the poset $(Z \times Z, \leq)$, …
    - is (5, 5) < (6, 4) ? YES
    - is (6, 5) < (6, 4) ? NO
    - is (3, 3) < (3, 3) ? NO

Lexicographic Order

- **Second step**: Extend the previous definition to the Cartesian product of n posets $(A_1, \leq_1)$, $(A_2, \leq_2)$, …, $(A_n, \leq_n)$:
- $(a_1, a_2, \dots, a_n) < (b_1, b_2, \dots, b_n)$ if $(a_1 <_1 b_1) \vee \exists i>0\ (a_1 = b_1, a_2 = b_2, \dots, a_i = b_i, a_{i+1} <_{i+1} b_{i+1})$
- $(a_1, a_2, \dots, a_n) \leq (b_1, b_2, \dots, b_n)$ if $(a_1 <_1 b_1) \vee \exists i>0\ (a_1 = b_1, a_2 = b_2, \dots, a_i = b_i, a_{i+1} <_{i+1} b_{i+1}) \vee (a_1 = b_1, a_2 = b_2, \dots, a_n = b_n)$

- Examples:
  - Is (1,1,1,2,1) < (1,1,1, 1, 2)? No
  - Is (1, 1, 1, 1, 1) < (1, 1, 1, 1, 2)? Yes

Hasse Diagram (哈斯图)

- Hasse diagram is a graphical display of a poset.
- A point is drawn for each element of the poset, and line segments are drawn between these points according to the following two rules:
  - 1. If $x < y$ in the poset, then the point corresponding to x appears lower in the drawing than the point corresponding to y.
  - 2. The line segment between the points corresponding to any two elements x and y of the poset is included in the drawing iff x covers y or y covers x.

Cover Relation

- Let $(S, \leq)$ be a poset. We say that an element $y \in S$ **covers** an element $x \in S$ if $x < y$ and there is no element $z \in S$ such that $x < z < y$. The set of pairs (x, y) such that y covers x is called **the covering relation** of $(S, \leq)$.

Hasse Diagrams

We produce Hasse Diagrams from directed graphs of relations by doing a **transitive reduction** plus a **reflexive reduction** (if weak) and (usually) **dropping arrowheads** (using, instead, "above" to give direction)

1) Transitive reduction — discard all arcs except those that "directly cover" an element.
2) Reflexive reduction — discard all self loops.

For $\supseteq$

$$\begin{tikzcd}
& \{a, b\} \arrow[loop above] & \\
\{a\} \arrow[ur] \arrow[loop left] & & \{b\} \arrow[ul] \arrow[loop right] \\
& \varnothing \arrow[ul] \arrow[uu] \arrow[ur] \arrow[loop below] &
\end{tikzcd}$$

$\equiv$

we write:

$$\begin{tikzcd}
& \{a, b\} & \\
\{a\} \arrow[ur, dash] & & \{b\} \arrow[ul, dash] \\
& \varnothing \arrow[ul, dash] \arrow[ur, dash] &
\end{tikzcd}$$

The Procedure Summary

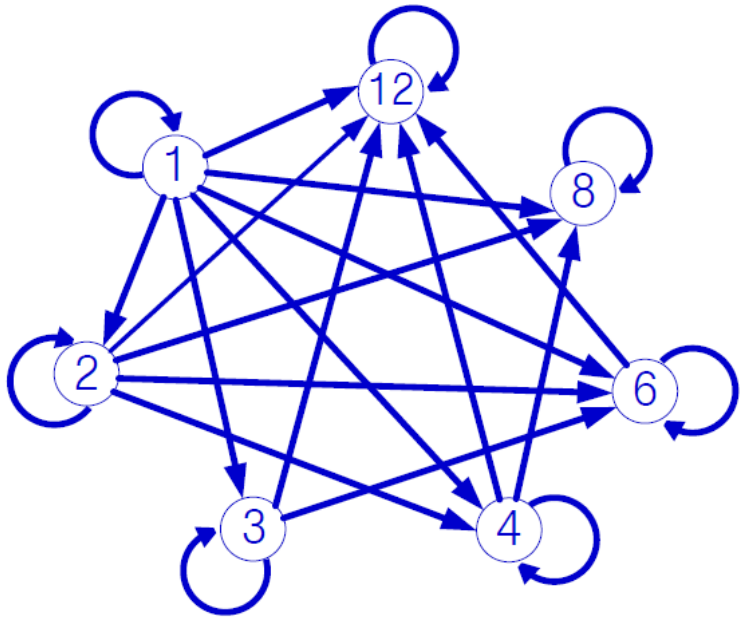
- Start with the directed graph for this relation.
- Because a partial ordering is reflexive, a loop (a, a) is present at every vertex a. Remove these loops.
- Next, remove all edges that must be in the partial ordering because of the presence of other edges and transitivity. That is, remove all edges (x, y) for which there is an element $z \in S$ such that $x < z$ and $z < x$.
- Finally, arrange each edge so that its initial vertex is below its terminal vertex (as it is drawn on paper). Remove all the arrows on the directed edges, because all edges point “upward” toward their terminal vertex.

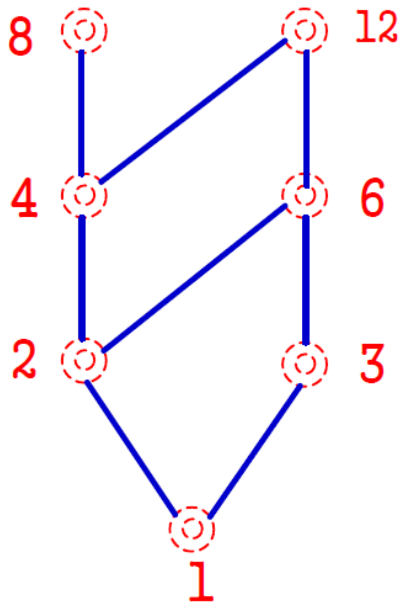
For $\supseteq$ we write:

$$\begin{tikzcd} & \{a, b\} \arrow[loop above] & \\ \{a\} \arrow[ur] \arrow[loop left] & & \{b\} \arrow[ul] \arrow[loop right] \\ & \varnothing \arrow[uu] \arrow[ul] \arrow[ur] \arrow[loop below] & \end{tikzcd} \equiv \begin{tikzcd}[arrows=dash] & \{a, b\} & \\ \{a\} \arrow[ur] & & \{b\} \arrow[ul] \\ & \varnothing \arrow[ul] \arrow[ur] & \end{tikzcd}$$

Hasse Diagram

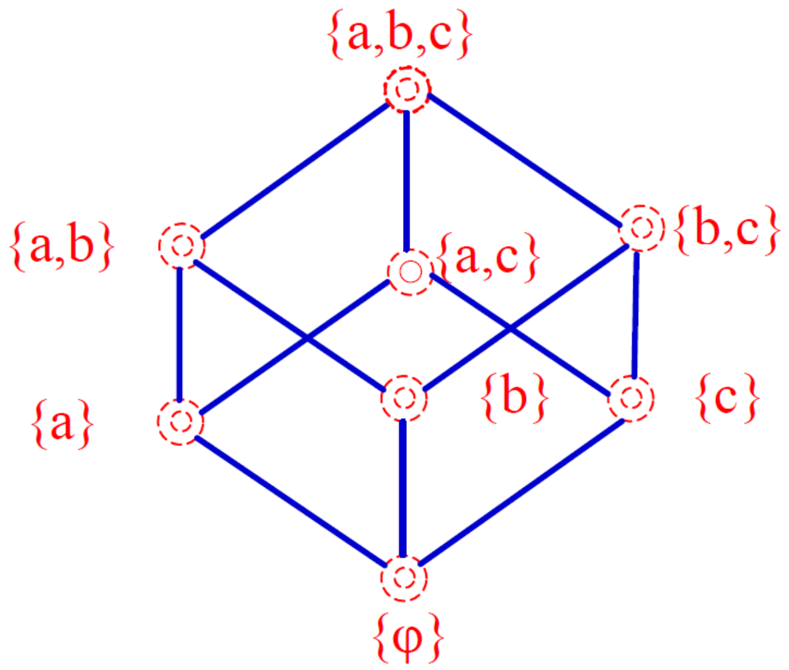
- **Example**: *A*={1,2,3,4,6,8,12}, integral division relation.





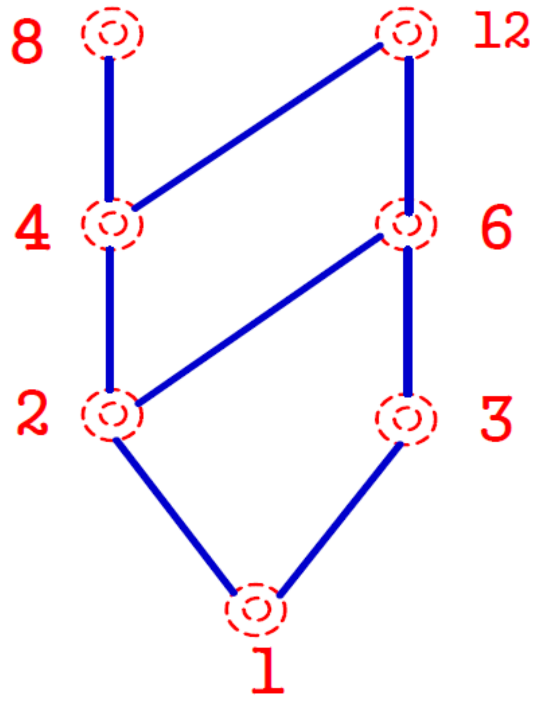
Hasse Diagram

- **Example**: S={a, b, c}, (P(S), ⊆)



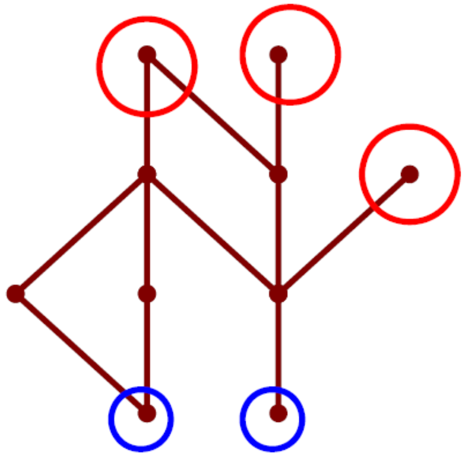
Maximum/Minimum/Greatest/Least

- Maximum/Minimum element
- 极大、极小
- Greatest/Least element
- 最大、最小
- Upper/Lower bound
- 上界、下界
- Least upper/Greatest lower bound
- 最小上界、最大下界



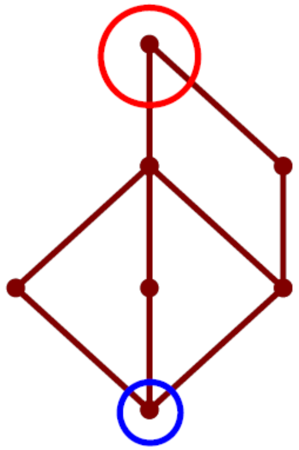
Minimum and Maximum

- **Definition**: In a poset S, an element z is a **minimum** element if there is no element b∈S, thus b ≤ z and b ≠ z.
- How about definition for **maximum** element?
- **Example**:
  - Reds are maximal.
  - Blues are minimal.



Least and Greatest

- **Definition**: In a poset S, an element z is a **Least** element if $\forall b \in S$, $z \leq b$.
- How about definition for **Greatest** element.
- **Example**:
  - Reds are greatest.
  - Blues are least.
- Greatest/Least may not exist.



Least and Greatest

- **Theorem**: In every poset, if the **greatest** element exists, then it is **unique**. Similarly for the **least**.
- **Proof**:
  - Suppose there are two greatest elements, a_1 and a_2, with $a_1 \neq a_2$. Then $a_1 \leq a_2$, and $a_2 \leq a_1$, by defn of greatest. So $a_1 = a_2$, a contradiction. Thus, our assumption was incorrect, and the greatest element, if it exists, is unique.
  - Similar proof for least.

Basics for Graphs

Key Points

- Graph basics and definitions
  - Vertices/nodes, edges, adjacency(邻域), incidence
  - Degree, in-degree, out-degree
  - Subgraphs, unions, isomorphism
  - Adjacency matrices
- Types of Graphs
  - Undirected graphs
    - Simple graphs, Multigraphs (多重边图), Pseudographs (伪图)
  - Digraphs, Directed multigraph
  - Bipartite (二部图)
  - Complete graphs, cycles (圈图), wheels (轮图), cubes, complete bipartite

Simple Graphs

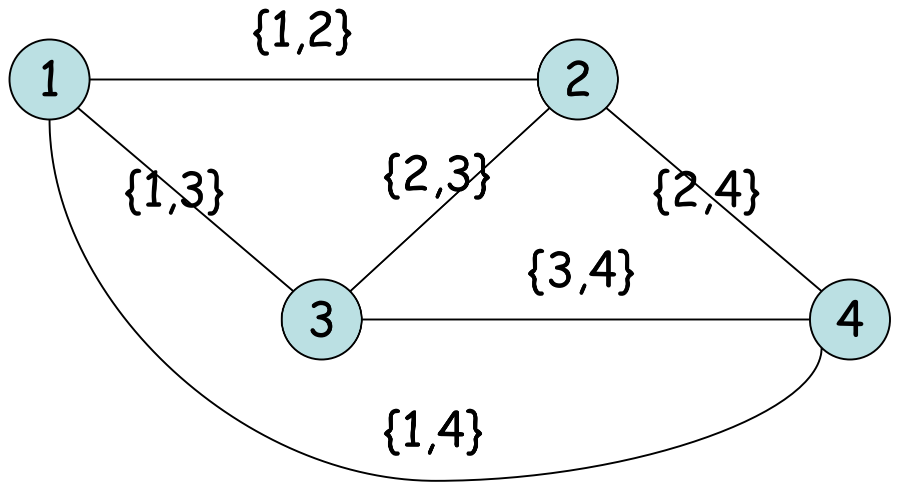
Different purposes require different types of graphs.

Exempli Gratia (EG):

Suppose a local computer network

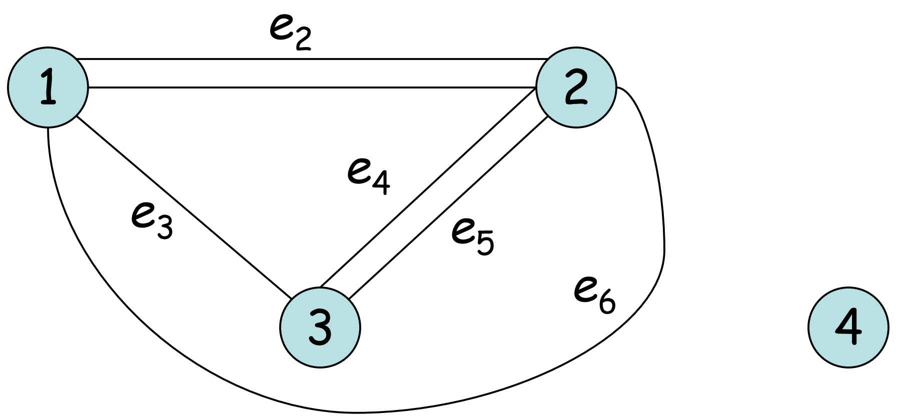
- Is bidirectional (undirected)
- Has no loops (no "self-communication")
- Has unique connections between computers

Sensible to represent as follows:



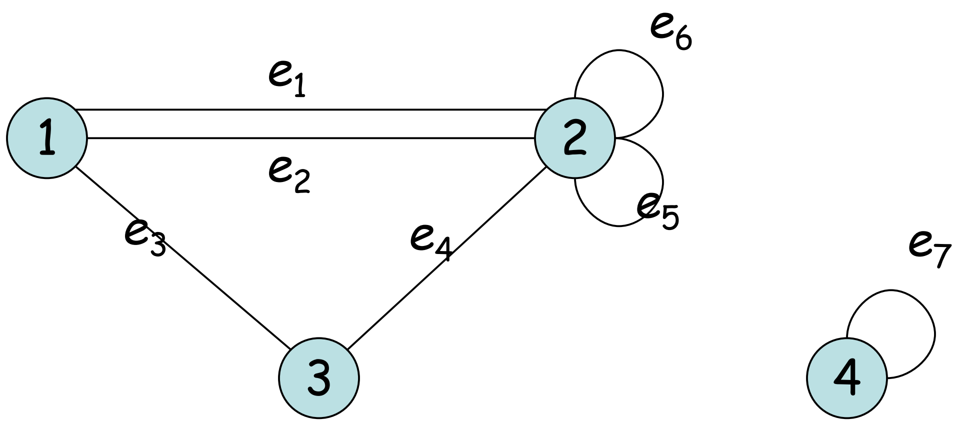
Multigraphs

If computers are connected via internet instead of directly, there may be several routes to choose from for each connection. Depending on traffic, one route could be better than another. Makes sense to allow multiple edges, but still no self-loops:



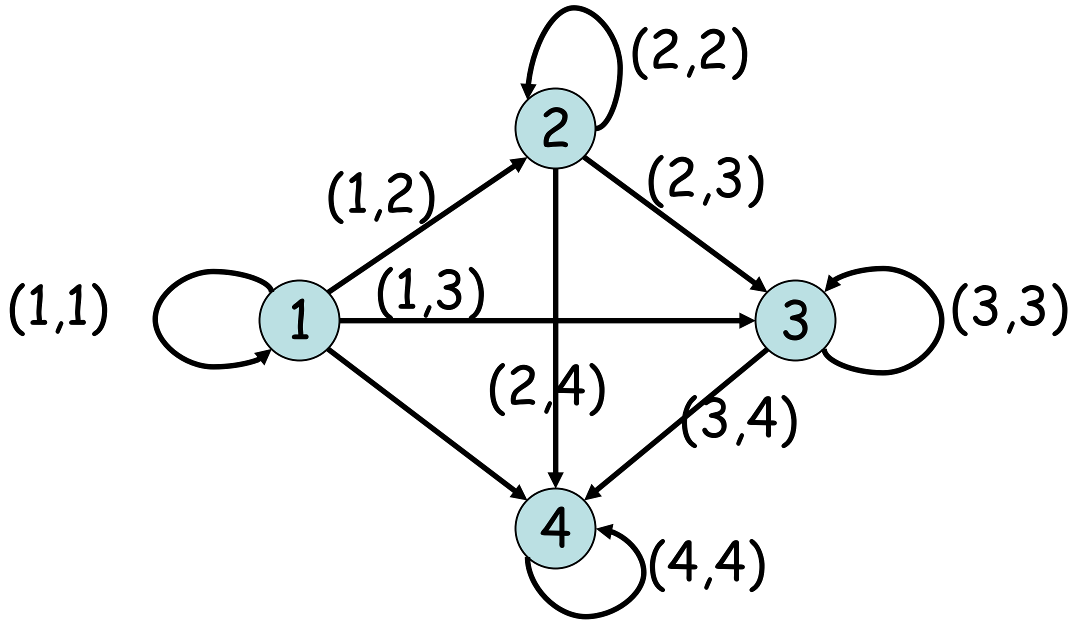
Pseudographs

Definition: A **pseudograph** G = (V,E,f) consists of a non-empty set V of **vertices** (or **nodes**), a set E (possibly empty) of **edges** and a function f with domain E and codomain the set of pairs and singletons in V.



Digraphs

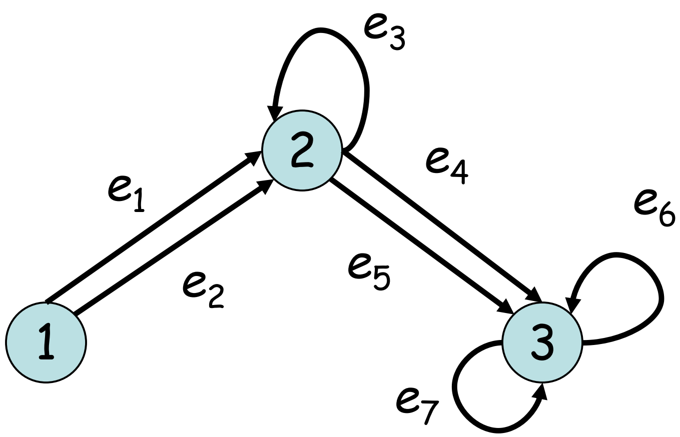
A: Each edge is directed so an ordered pair (or tuple) rather than unordered pair.



Thus the set of edges E is just the represented relation on V.

Directed Multigraphs

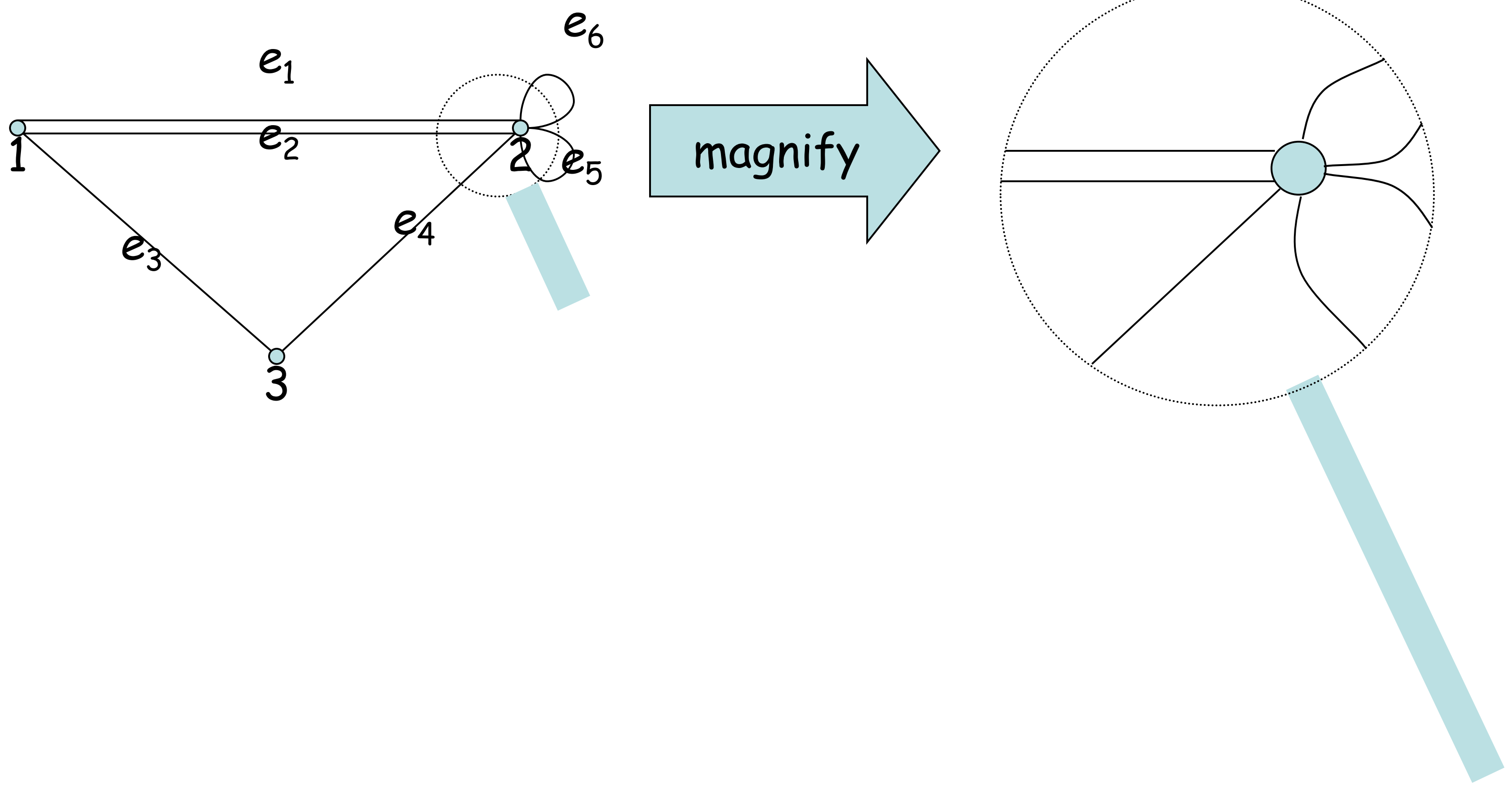
A: Have function with domain the edge set and codomain $V \times V$.



$e_1 \rightarrow (1,2)$, $e_2 \rightarrow (1,2)$, $e_3 \rightarrow (2,2)$, $e_4 \rightarrow (2,3)$,
$e_5 \rightarrow (2,3)$, $e_6 \rightarrow (3,3)$, $e_7 \rightarrow (3,3)$

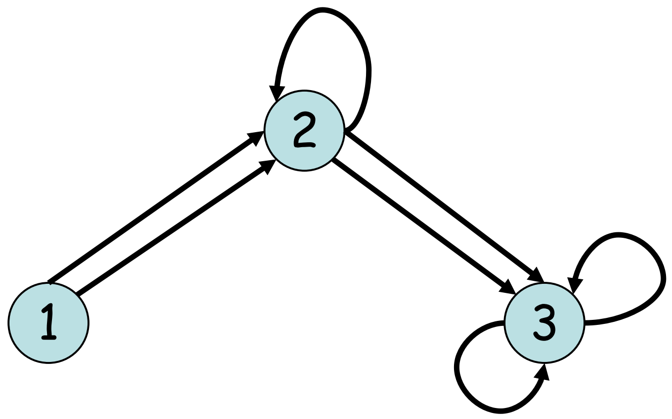
Degree

A: Add 1 for every regular edge incident with vertex and 2 for every loop. Thus deg(2) = 1 + 1 + 1 + 2 + 2 = 7



Oriented Degree when Edges Directed

The **in-degree** of a vertex (deg^-) counts the number of edges that stick in to the vertex. The **out-degree** (deg^+) counts the number sticking out.



Q: What are in-degrees and out-degrees of all the vertices?

Handshaking Theorem*

Theorem: In an undirected graph $$|E| = \frac{1}{2}\sum_{v\in V}\deg(v)$$

In a directed graph $$|E| = \sum_{v\in V}\deg^{+}(v) = \sum_{v\in V}\deg^{-}(v)$$

Q: In a party of 5 people can each person be friends with exactly three others?

A: Imagine a simple graph with 5 people as *vertices* and edges being undirected edges between friends (simple graph assuming friendship is symmetric and irreflexive). Number of friends each person has is the degree of the person.

Handshaking would imply that
$|E|$ = (sum of degrees)/2 or
$2|E|$ = (sum of degrees) = $(5\cdot3) = 15$.
Impossible as 15 is not even. In general:

Adjacency Matrix-Directed Multigraphs

Can easily generalize to directed multigraphs by putting in the number of edges between vertices, instead of only allowing 0 and 1:

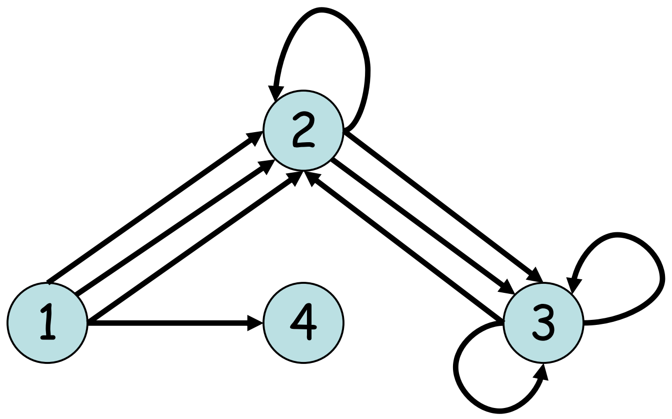
For a directed multigraph $G = (V,E)$ define the matrix A_G by:

- Rows, Columns –one for each vertex in V
- Value at i th row and j th column is the number of edges with source the

i th vertex and target the j th vertex

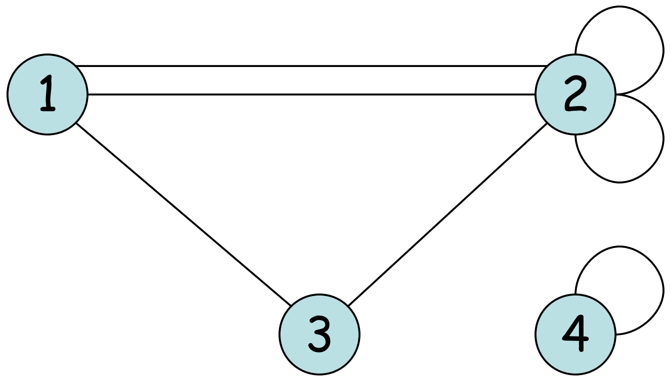
Adjacency Matrix-Directed Multigraphs

A:



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

Adjacency Matrix-General



A:

Notice that answer is *symmetric.*

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

Graph Isomorphism Undirected Graphs

Definition: Suppose $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are pseudographs. Let f $: V_1 \rightarrow V_2$ be a function s.t.:

1. f is bijective
2. for all vertices u,v in V_1, the number of edges **between u and v** in G_1 is the same as the number of edges between f (u) and f (v) in G_2.

Then f is called an **isomorphism** and G_1 is said to be **isomorphic** to G_2.

Graph Isomorphism Digraphs

Definition: Suppose $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are directed multigraphs. Let $f : V_1 \rightarrow V_2$ be a function s.t.:

1) f is bijective
2) for all vertices u,v in V_1, the number of edges **from u to v** in G_1 is the same as the number of edges between f (u) and f (v) in G_2.

Then f is called an **isomorphism** and G_1 is said to be **isomorphic** to G_2.

Note: Only difference between two definitions is the italicized "from" in no. 2 (was "between").

Properties of Isomorphims

Since graphs are completely defined by their vertex sets and the number of edges between each pair, isomorphic graphs must have the same intrinsic properties. I.e. isomorphic graphs have the same…

…number of vertices and edges

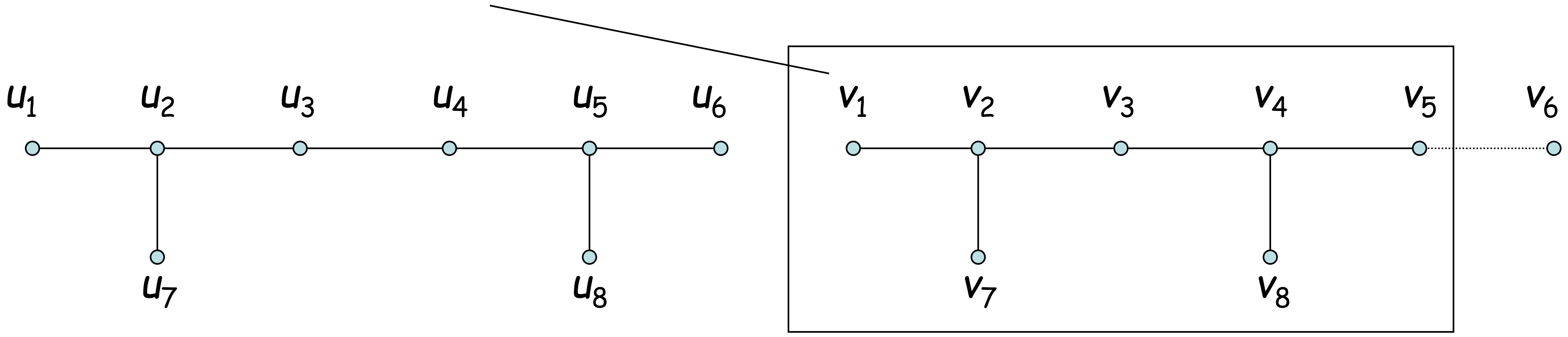
…degrees at corresponding vertices

…types of possible subgraphs

…any other property defined in terms of the basic graph theoretic building blocks!

Graph Isomorphism-Negative Examples

This subgraph is not a subgraph of the left graph.



Why not? Deg. 3 vertices must map to deg. 3 vertices. Since subgraph and left graph are symmetric, can assume v_2 maps to u_2. Adjacent deg. 1 vertices to v_2 must map to degree 1 vertices, forcing the deg. 2 adjacent vertex v_3 to map to u_3. This forces the other vertex adjacent to v_3, namely v_4 to map to u_4. But then a deg. 3 vertex has mapped to a deg. 2 vertex$\rightarrow\leftarrow$ •

Paths and Connectivity

Paths

A: For simple graphs, any edge is unique between vertices so listing the vertices gives us the edge-sequence as well.

Definition: A **simple path** contains no duplicate edges (though duplicate vertices are allowed). A **cycle** (or **circuit**) is a path which starts and ends at the same vertex.

Note: Simple paths need not be in simple graphs. E.g., may contain loops.

Paths in Directed Graphs

One can define paths for directed graphs by insisting that the target of each edge in the path is the source of the next edge:

Definition: A **path** of length n in a directed graph is a sequence of n edges e_1, e_2, … ,e_n such that the target of e_i is the source e_{i+1} for each i.

In a digraph, one may instead define a path of length n as a sequence of n+1 vertices v_0, v_1, v_2, … ,v_n such that for each consecutive pair v_i , v_{i+1} there is an edge from v_i to v_{i+1} .

Paths in Directed Graphs

A:

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

1. $1 \rightarrow 3 \rightarrow 2 \rightarrow 4$.

2. There's no path from 4 to 1. From 4 must go to 2, from 2 must stay at 2 or return to 4. In other words 2 and 4 are *disconnected* from 1.

Connectivity

Definition: Let G be a pseudograph. Let u and v be vertices. u and v are **connected** to each other if there is a path in G which starts at u and ends at v. G is said to be **connected** if all vertices are connected to each other.

1. Note: Any vertex is automatically connected to itself via the empty path.
2. Note: A suitable definition for directed graphs will follow later.

Connectivity in Directed Graphs

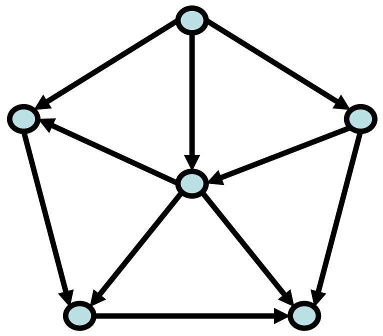
Resolution: Don't bother choosing which definition is better. Just define to separate concepts:

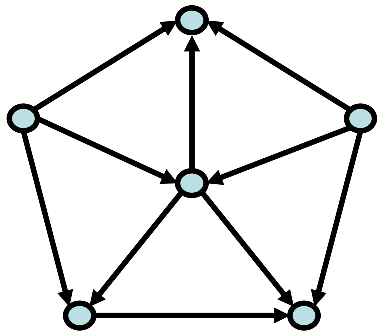
1. **Weakly connected**: can get from a to b in underlying undirected graph
2. **Semi-connected** (my terminology): can get from a to b OR from b to a in digraph
3. **Strongly connected**: can get from a to b AND from b to a in the digraph

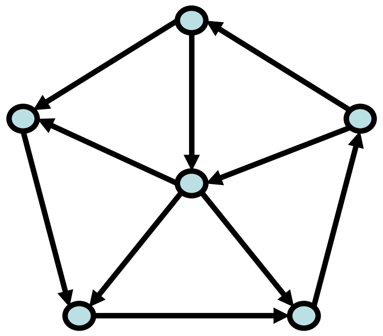
Definition: A graph is **strongly** (resp. **semi**, resp. **weakly**) connected if every pair of vertices is connected in the same sense.

Connectivity in Directed Graphs

Q: Classify the connectivity of each graph.



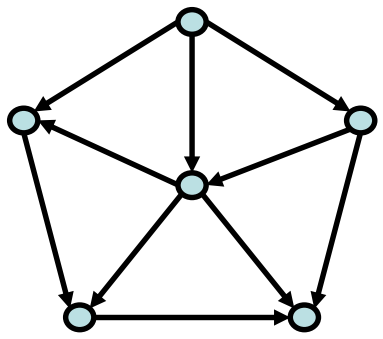


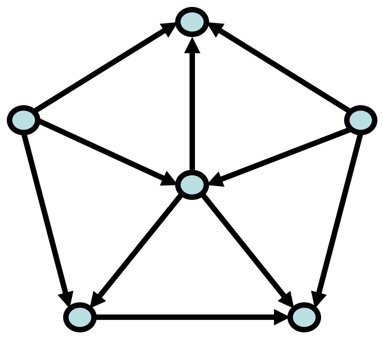


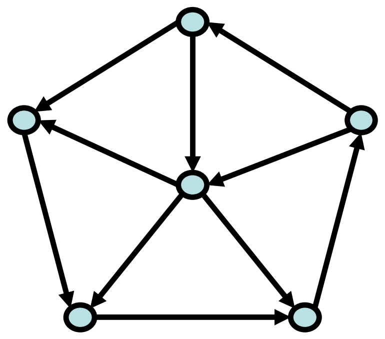
Connectivity in Directed Graphs

A:

semi *weak* *strong*



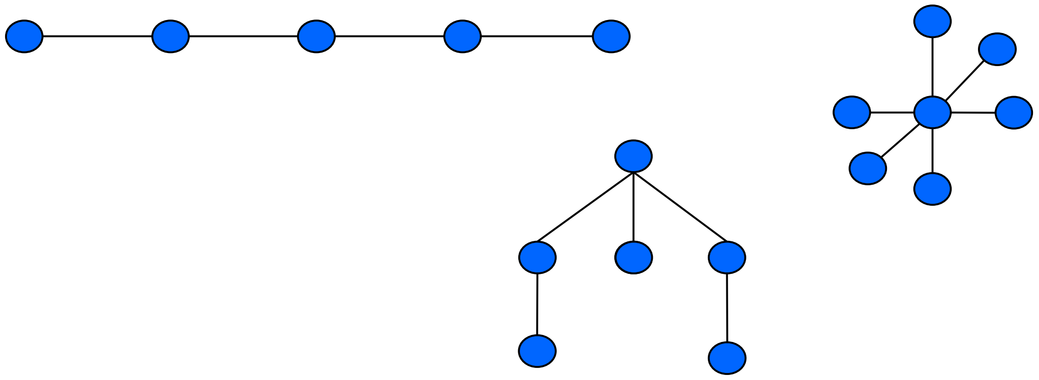




Tree

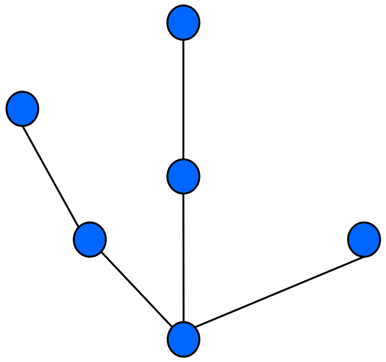
Graphs with no cycles?

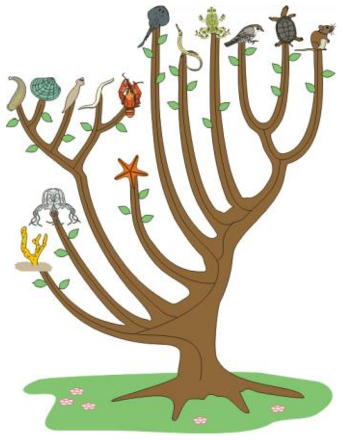
A forest.



Connected graphs with no cycles?

A tree.





Tree Characterizations

Definition. A tree is a connected graph with no cycles.

Characterization by paths:

A graph is a tree if and only if

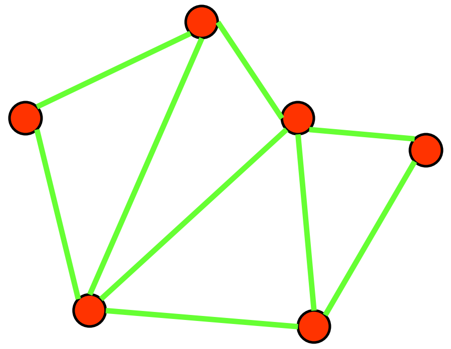
there is a unique simple path between every pair of vertices.

Characterization by number of edges:

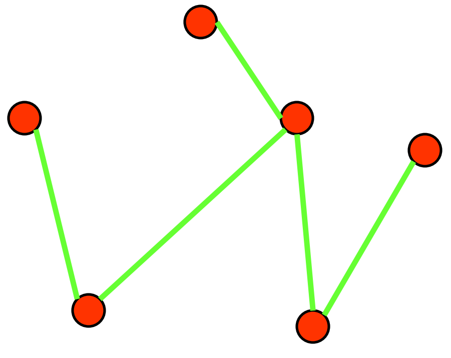
A graph is a tree if and only if it is connected and has n-1 edges.

Trees

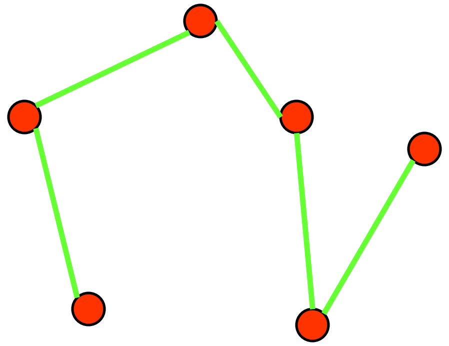
- **Example:** Are the following graphs trees?



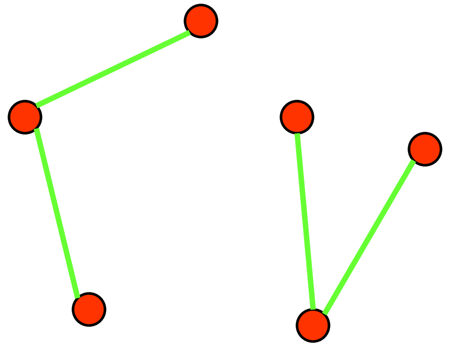
No.



Yes.



Yes.



No.

Spanning Trees

•**Definition:** Let G be a simple graph. A spanning tree of G is a subgraph of G that is a tree containing every vertex of G.

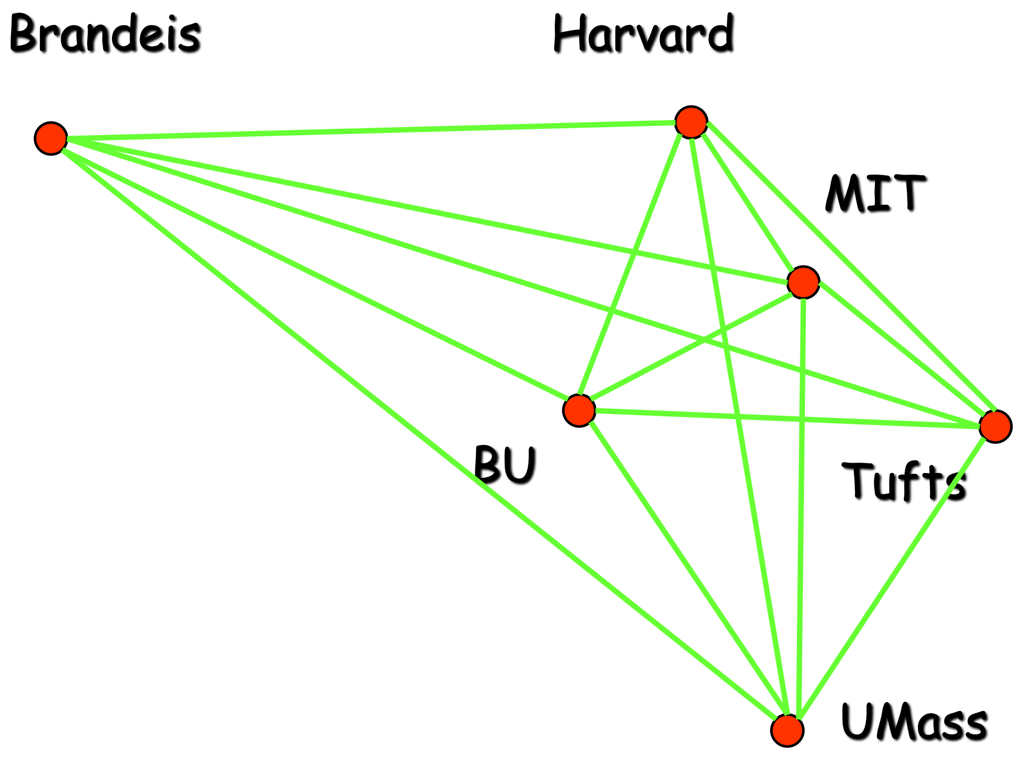
•**Note:** A spanning tree of G = (V, E) is a connected graph on V with a minimum number of edges
(|V| - 1).

•**Example:** Since winters in Boston can be very cold, six universities in the Boston area decide to build a tunnel system that connects their libraries.

- A spanning tree in an undirected graph G(V,E) is a subset of edges $T \subseteq E$ that are acyclic and connect all the vertices in V.
- A spanning tree must consist of exactly n-1 edges.
- Suppose that each edge has a weight associated with it. Say that the weight of a tree T is the sum of the weights of its edges $w(T) = \sum_{e \in T} w(e)$
- The minimum spanning tree in a weighted graph G(V,E) is one which has the smallest weight among all spanning trees in G(V,E)

Spanning Trees

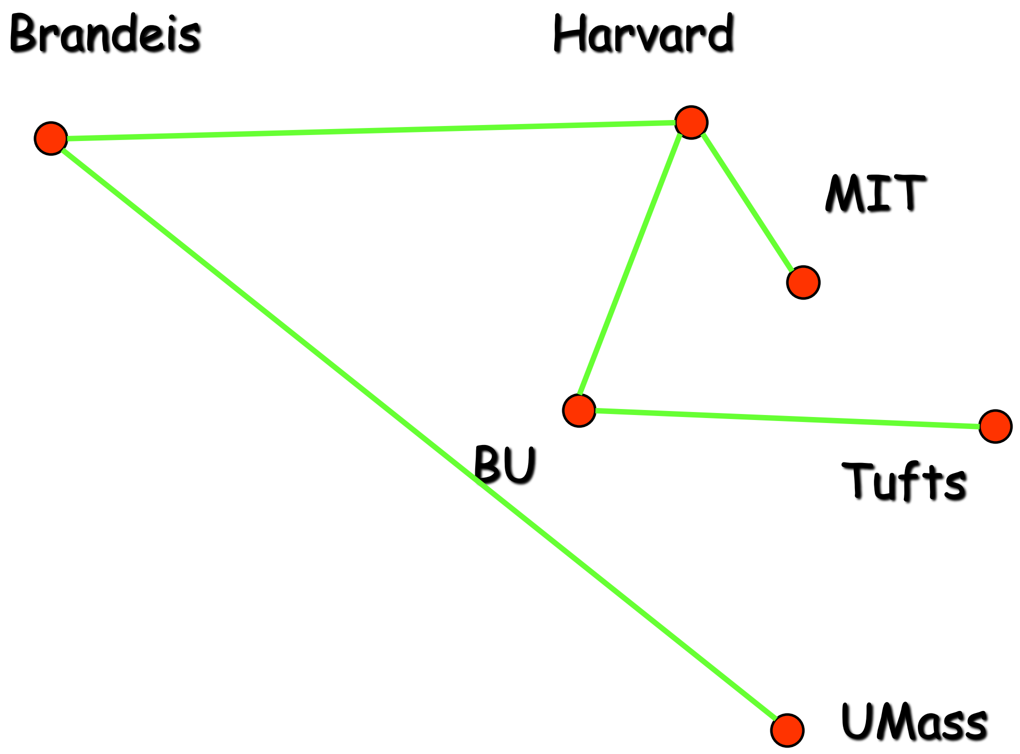
- The complete graph including all possible tunnels:



The spanning trees of this graph connect all libraries with a minimum number of tunnels.

Spanning Trees

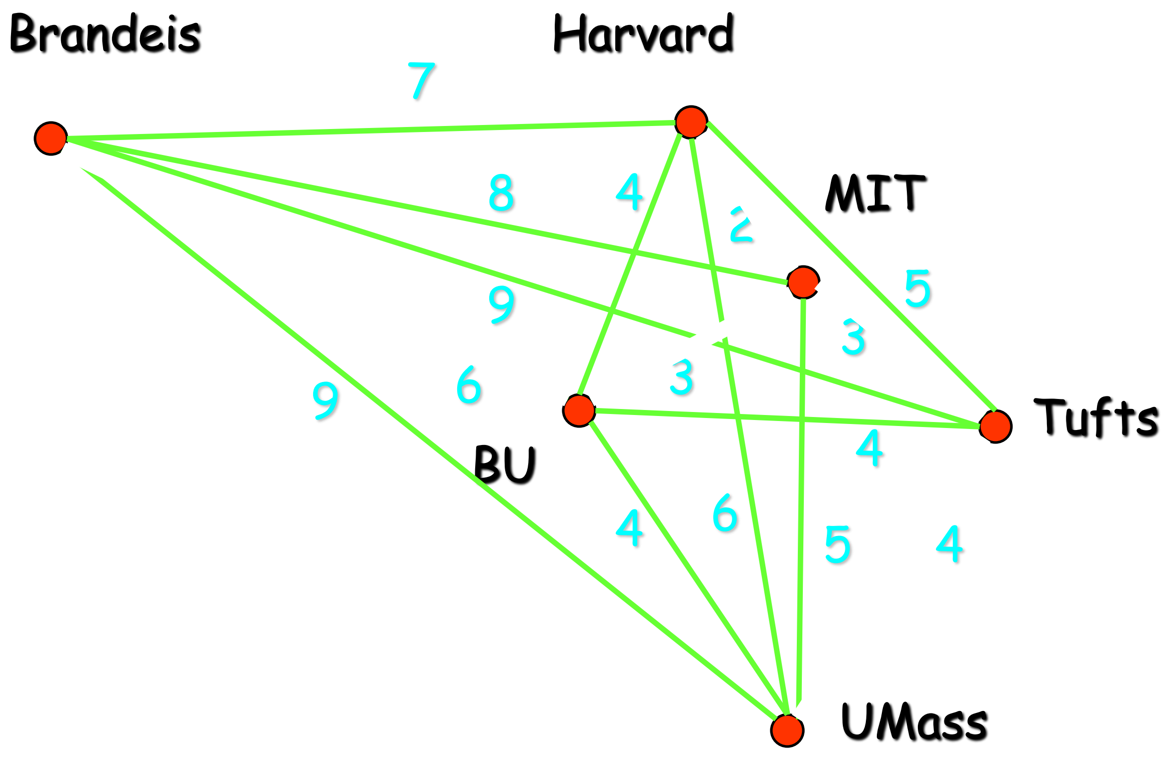
- Example for a spanning tree:



Since there are 6 libraries, 5 tunnels are sufficient to connect all of them.

Spanning Trees

- The complete graph with cost labels (in billion $):



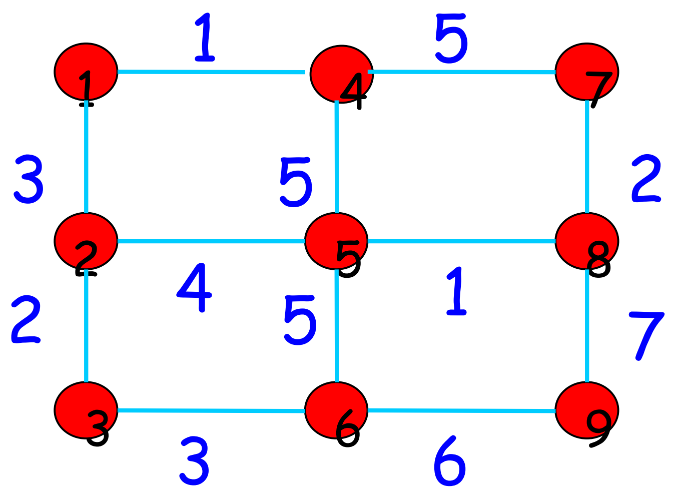
The least expensive tunnel system costs $20 billion.

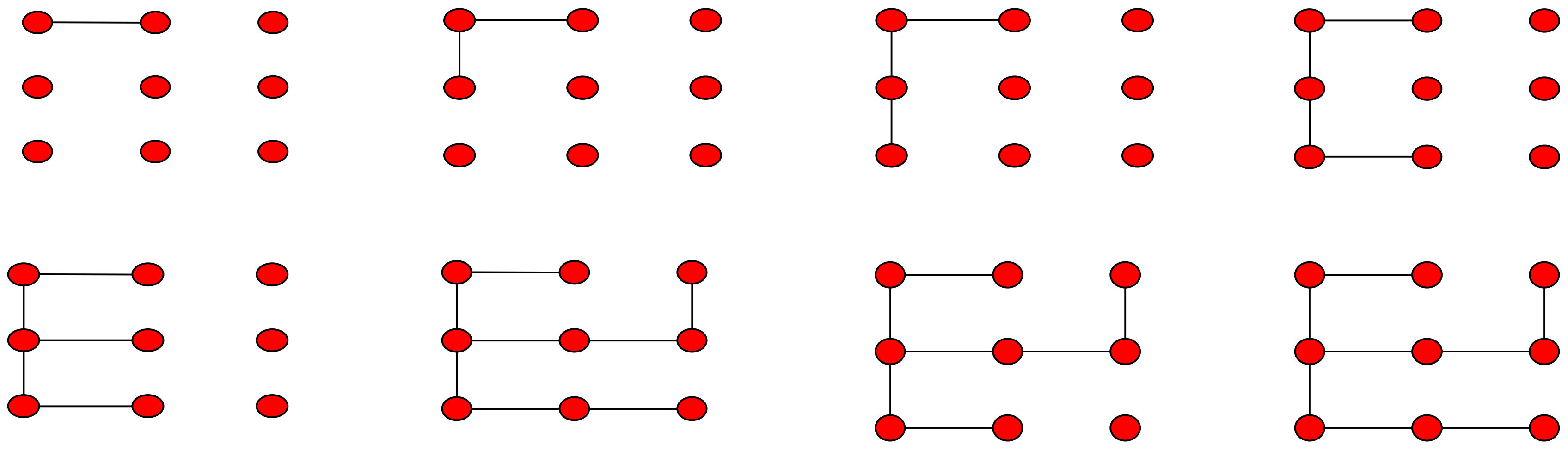
Spanning Trees

•Prim's Algorithm:

- Begin by choosing any edge with **smallest weight** and putting it into the spanning tree,
- successively add to the tree edges of **minimum weight** that are incident to a vertex already in the tree and not forming a simple circuit with those edges already in the tree,
- stop when (n – 1) edges have been added.

Prim's algorithm





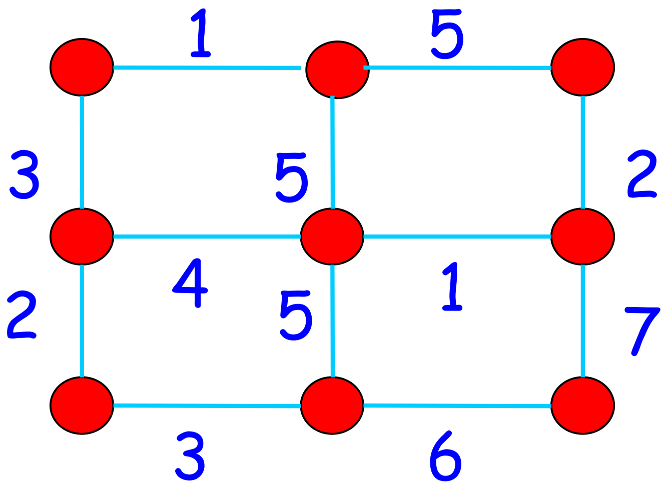
Spanning Trees

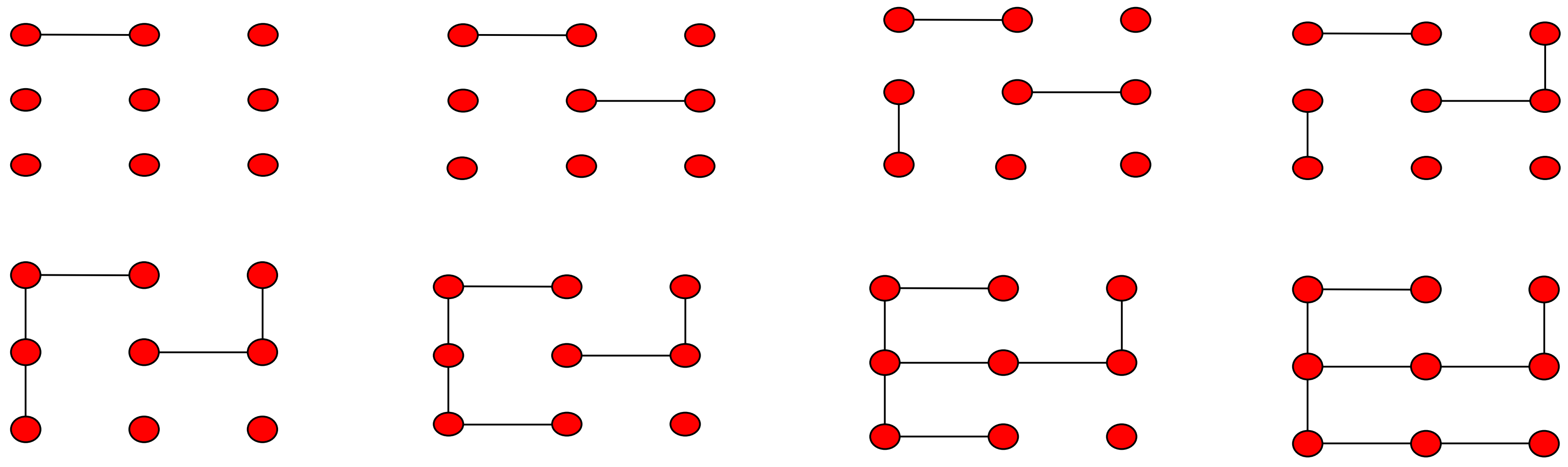
•**Kruskal's Algorithm:**

•Kruskal's algorithm is identical to Prim's algorithm, except that it does not demand new edges to be incident to a vertex already in the tree.

•Both algorithms are **guaranteed** to produce a minimum spanning tree of a connected weighted graph.

Kruskal's algorithm





Rooted Trees

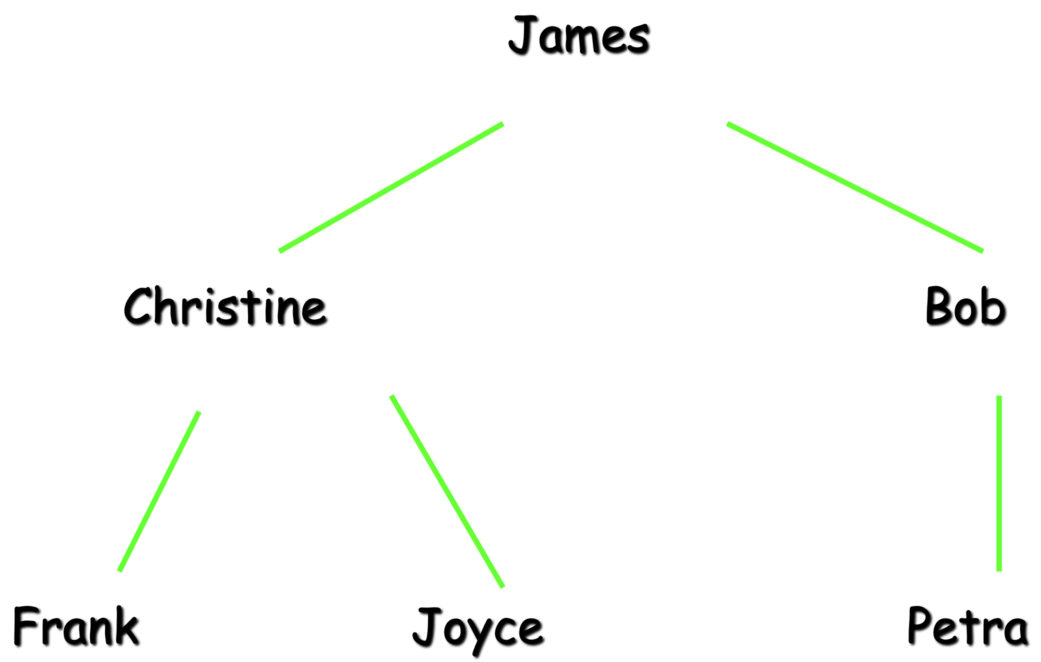
- We often designate a particular vertex of a tree as the **root**. Since there is a unique path from the root to each vertex of the graph, we direct each edge away from the root.
- Thus, a tree together with its root produces a **directed graph** called a **rooted tree**.

- If v is a vertex in a rooted tree other than the root, the **parent** of v is the unique vertex u such that there is a directed edge from u to v.
- When u is the parent of v, v is called the **child** of u.
- Vertices with the same parent are called **siblings**.
- The **ancestors** of a vertex other than the root are the vertices in the path from the root to this vertex, excluding the vertex itself and including the root.

Rooted Trees

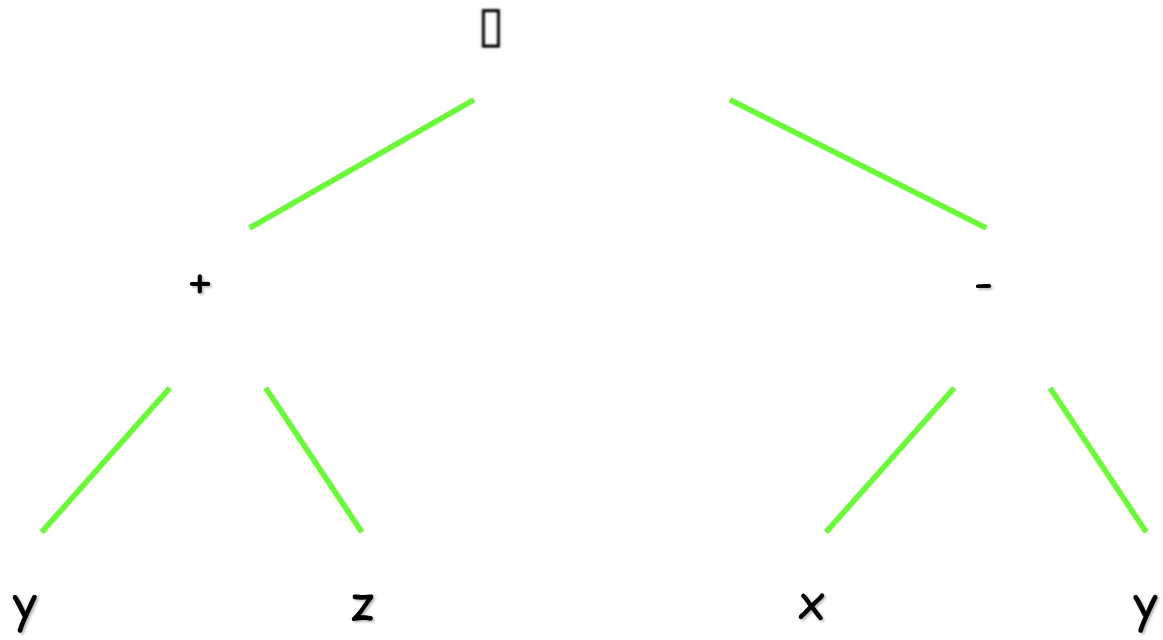
- The **level** of a vertex v in a rooted tree is the length of the unique path from the root to this vertex.
- The level of the root is defined to be zero.
- The **height** of a rooted tree is the maximum of the levels of vertices.

- **Example I:** Family tree



Trees

- **Example III:** Arithmetic expressions



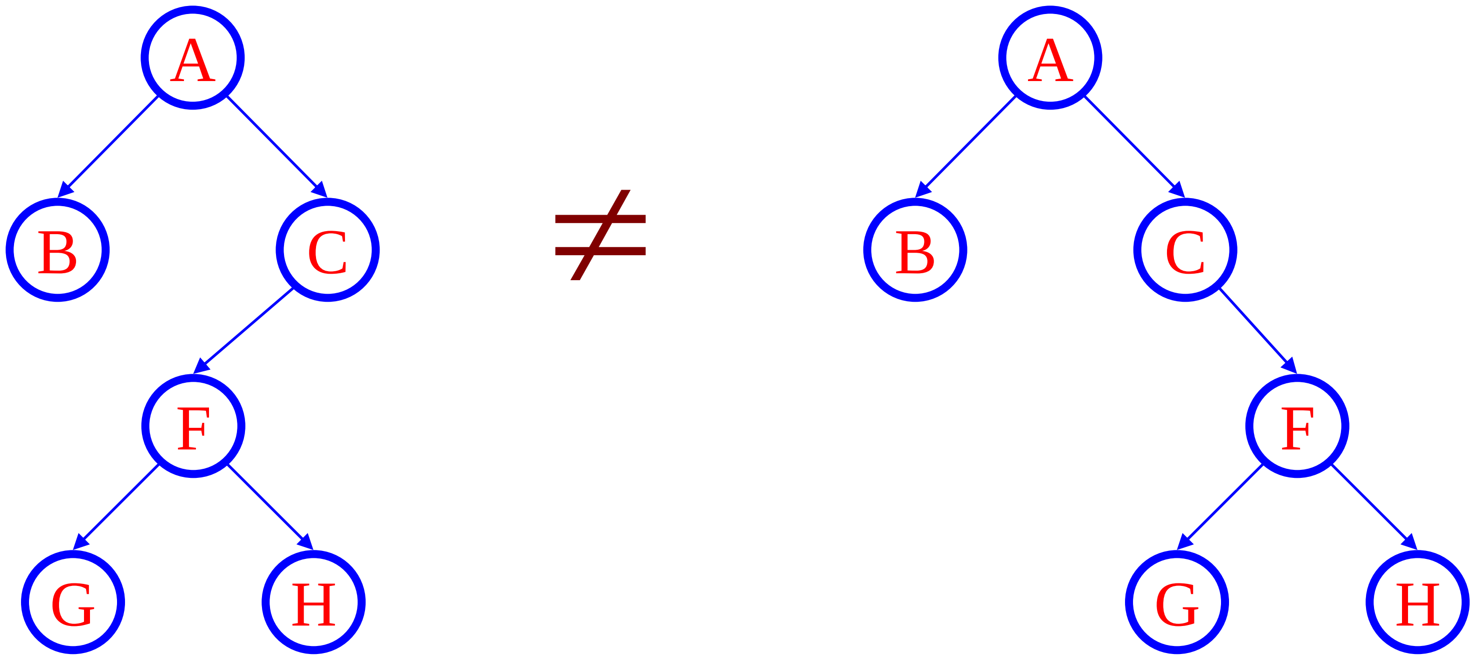
This tree represents the expression (y + z)   (x - y).

Binary Trees

- Every node has at most two children
- Most popular tree in computer science
- Given N nodes, what is the minimum depth of a binary tree?
- What is the maximum depth of a binary tree with N nodes?

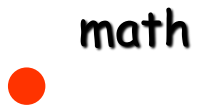
Binary Trees

- Notice:
- we distinguish between left child and right child



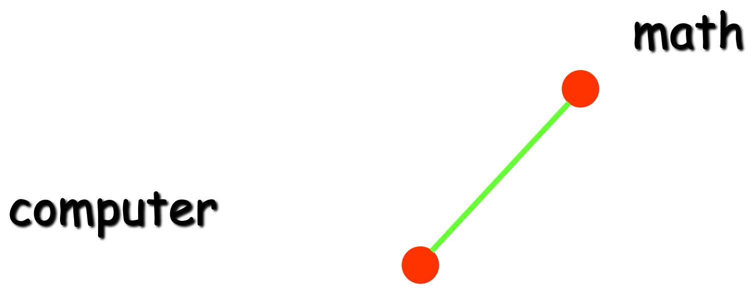
Binary Search Trees

- **Example:** Construct a binary search tree for the strings **math, computer, power, north, zoo, dentist, book**.



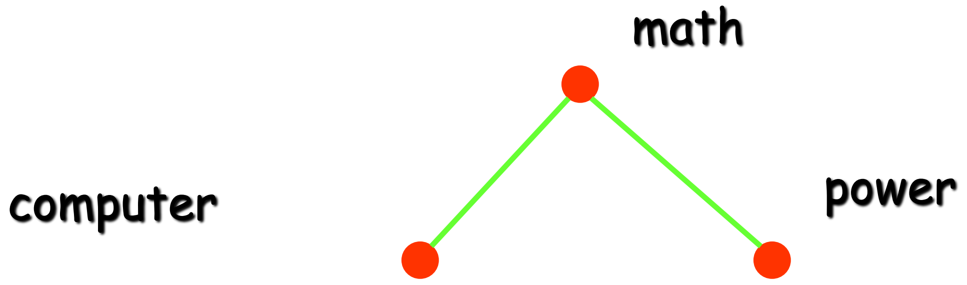
Binary Search Trees

•**Example:** Construct a binary search tree for the strings **math, computer, power, north, zoo, dentist, book**.



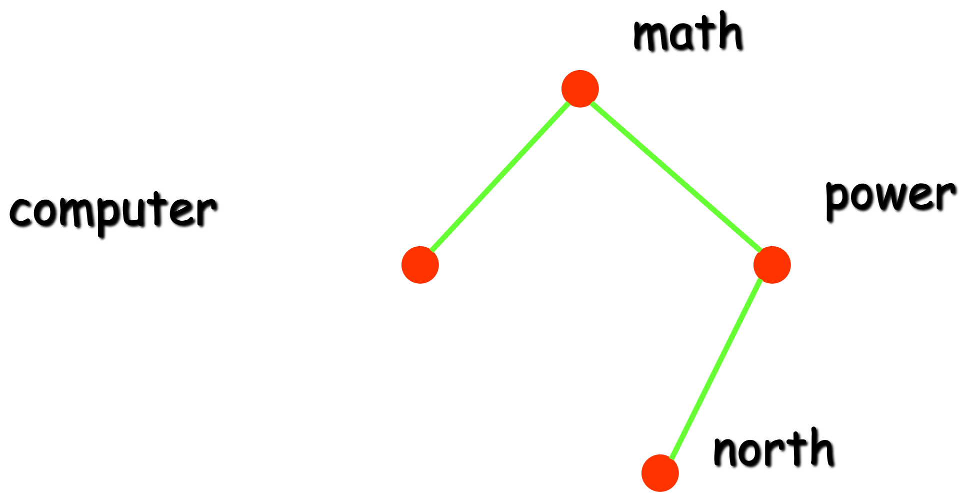
Binary Search Trees

- **Example:** Construct a binary search tree for the strings **math, computer, power, north, zoo, dentist, book**.



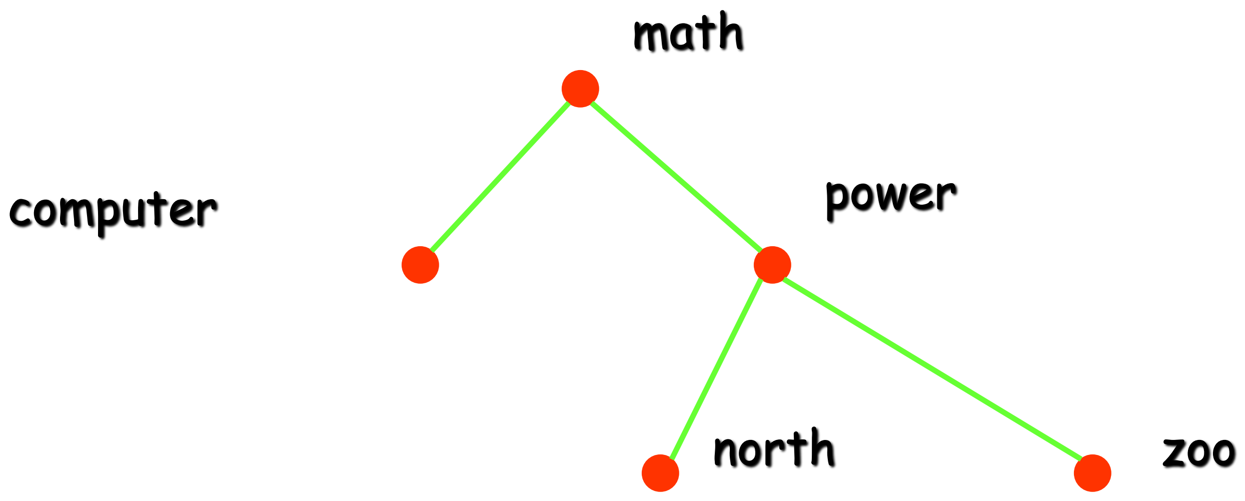
Binary Search Trees

- **Example:** Construct a binary search tree for the strings **math, computer, power, north, zoo, dentist, book**.



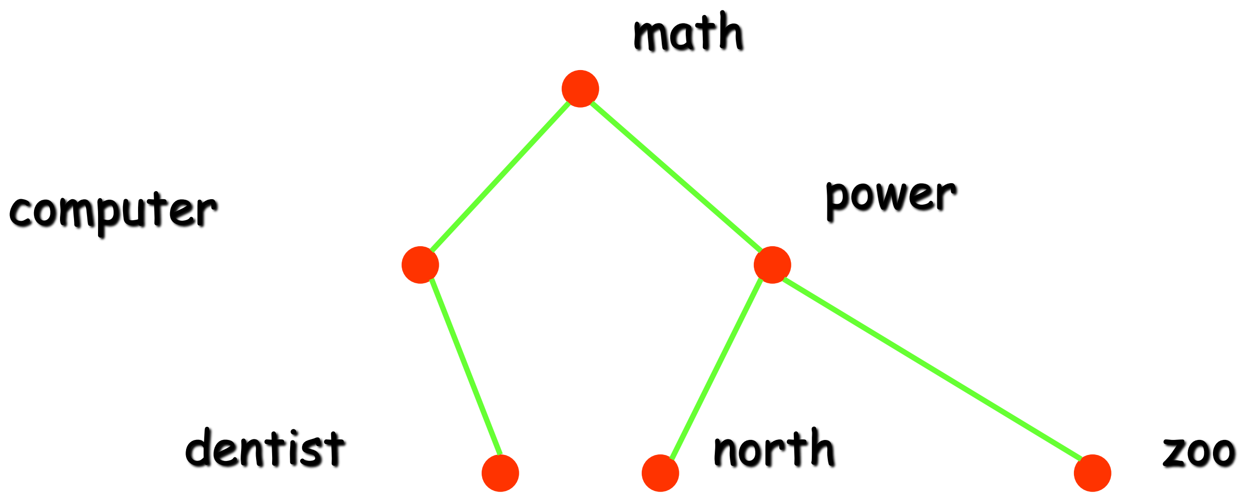
Binary Search Trees

- **Example:** Construct a binary search tree for the strings **math, computer, power, north, zoo, dentist, book**.



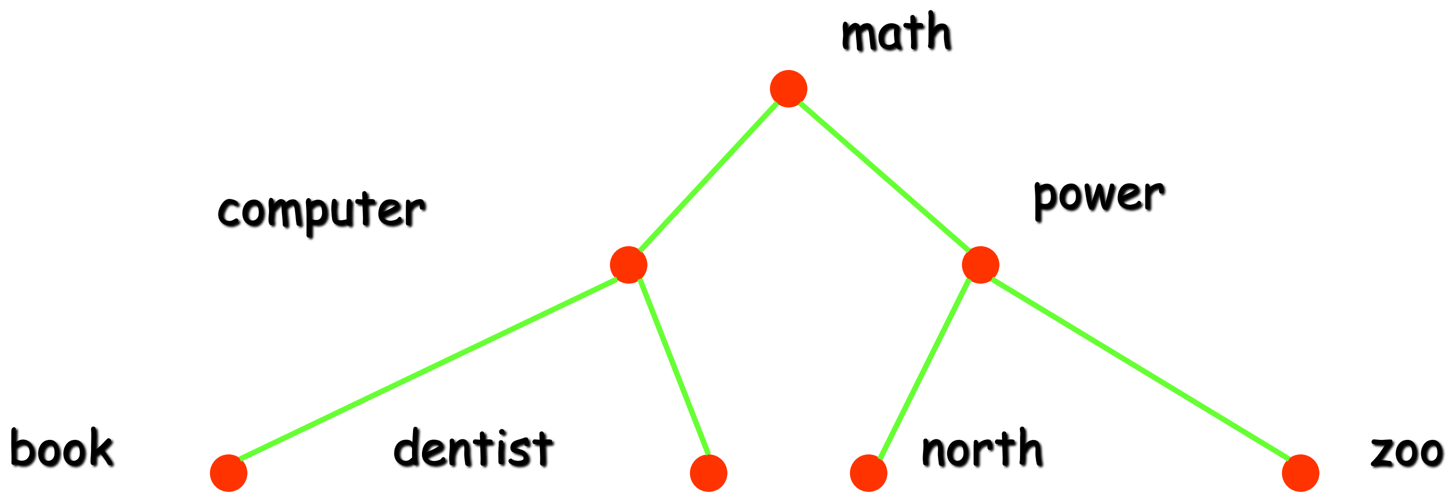
Binary Search Trees

- **Example:** Construct a binary search tree for the strings **math, computer, power, north, zoo, dentist, book**.



Binary Search Trees

- **Example:** Construct a binary search tree for the strings **math, computer, power, north, zoo, dentist, book**.



Binary Search Trees

- To perform a search in such a tree for an item x, we can start at the root and compare its key to x. If x is **less** than the key, we proceed to the **left** child of the current vertex, and if x is **greater** than the key, we proceed to the **right** one.
- This procedure is repeated until we either found the item we were looking for, or we cannot proceed any further.