Massachusetts Institute of Technology
6.042J/18.062J, Fall '05: Mathematics for Computer Science
Prof. Albert R. Meyer and Prof. Ronitt Rubinfeld

Course Notes, Week 4
September 26
revised October 17, 2005, 463 minutes

Binary Relations

1 Are We Related?

Questions about how two things are related are bound to come up whatever you're doing. For two people, you might ask if they're related (as family), if they know each other, if one is older than the other, if they're the same sex, race, age,. . . . For two countries, you might ask if they trade with each other, if the first has a higher per capita income, if a visa is required to visit one from the other,. . . . In Mathematics or Computer Science, if two variables are assigned values, we're used to asking if the values are the same, if the first value is bigger than the second (assuming both values are real numbers), if the values have a common divisor (assuming both values are integers), if the first value is a member of the second (assuming the second value is a set), if the first value is the domain of the second (assuming the first is a set and the second is a function). These are all examples of *binary relations*.

The concept of binary relation is as fundamental mathematically as the concept of function or set. In these Notes we'll define some basic terminology for binary relations, and then we'll focus on two especially important kinds of binary relations: *equivalence relations* and *partial orders*.

1.1 Relations and Functions

Here's the official definition:

Definition 1.1. A *binary relation*, R, consists of a set, A, called the *domain* of R, a set, B, called the *codomain* of R, and a subset of $A \times B$ called the *graph* of R.

For example, we can define an "is teaching relation" for Fall '05 at MIT to have domain equal to the names of all the teaching staff (faculty, T.A.'s, *etc.*) and codomain equal to all the subject numbers in the current catalogue. Its graph would look like

$$\begin{aligned}&\{(\texttt{Albert R. Meyer}, 6.042), (\texttt{David Shin}, 18.062), (\texttt{Sayan Mitra}, 6.04),\\ &\ (\texttt{Albert R. Meyer}, 18.062), (\texttt{Charles E. Leiserson}, 6.046),\\ &\ (\texttt{Donald Sadoway}, 3.091), \dots\}\end{aligned}$$

Notice that Definition 1.1 is exactly the same as the definition of a *function*, except that it doesn't require the functional condition that, for each domain element, a, there is *at most* one pair in the graph whose first coordinate is a. So a function is a special case of a binary relation.

A relation whose domain is A and codomain is B is said to be "between A and B", or "from A to B." When the domain and codomain are the same set, A, we simply say the relation is "on A." It's common to use infix notation "$a\,R\,b$" to mean that the pair (a, b) is in the graph of R.

1.2 Images and Inverse Images

Before we go any further, it's worth introducing some notation that we'll get a lot of mileage out of. If R is a binary relation from A to B, and C is any set, define

$$\begin{aligned} CR &::= \{b \in B \mid cRb \text{ for some } c \in C\}, \\ RC &::= \{a \in A \mid aRc \text{ for some } c \in C\}. \end{aligned}$$

The set CR is called the *image* of C under R. Notice that if R happened to be a function, the notation $R(C)$ from Week 3 Notes would also describe the image of C under R.

The set RC is called the *inverse image* of C under R. Notice the clash in notation when R happens to be a function: $R(C) = CR$, not RC. Sorry about that.

1.3 Surjective and like that

A relation with the property that every codomain element is related to some domain element is called a *surjective* (or *onto*) relation —again, the same definition as for functions. More concisely, a relation, R, between A and B is surjective iff $AR = B$. Likewise, a relation with the property that every domain element is related to some codomain element is called a *total* relation; more concisely, R is total iff $A = RB$.

The Fall '05 "is teaching relation" relation above is not surjective since none of the Spring term-only subjects are being taught. It's not total either, since not all the eligible teaching staff are actually teaching this term.

2 Equivalence Relations

An equivalence relation on a set of objects comes about when all we care about is some property—say the size, shape, or color—of the objects rather than the objects themselves. We say two objects with the same property value are "equivalent." Of course this happens all the time, which is why equivalence relations appear everywhere.

For example, two triangles in the plane are *congruent* iff they have the same three lengths of sides. They are *similar* iff they have the same three sizes of angles.

Representation-equivalence comes up in Computer Science as the relation between representations of the same abstract data type. For example, the simplest way of representing a finite set of numbers is as an unsorted list. The two lists `(3 4 -2 177 5)` and `(177 -2 3 5 4)` are "representation-equivalent" because they represent the same set.

2.1 Equivalence by Function

Abstractly, we assume there is some function that extracts the angles, size, color, or whatever other property of elements we're interested in. Two elements would be considered equivalent iff the function extracts the same value for each.

For example, if f_c is the function mapping a triangle to the lengths its sides, then f_c determines the congruence relation. If f_s is the function mapping a triangle to the sizes of its angles, then f_s determines the similarity relation.

Definition 2.1. Given any total function, f, with domain A, define the binary relation $\equiv_f$ on A by the rule:

$$a \equiv_f b \quad \text{iff} \quad f(a) = f(b) \tag{1}$$

for all $a, b \in A$.

A binary relation is an *equivalence relation* iff it equals $\equiv_f$ for some f.

So congruence of triangles is an equivalence relation because it is $\equiv_{f_c}$, as is triangle similarity because it is $\equiv_{f_s}$. Likewise representation-equivalence on number lists is an equivalence relation because it is $\equiv_{f_r}$, where f_r maps a representation to the set it represents.

Quick exercise: Show that the equality relation on elements of a set, A, is actually an equivalence relation according to Definition 2.1 by describing an $I : A \to A$ such that equality is $\equiv_I$.

Congruence modulo n is another equivalence that we will explore in detail when we introduce elementary number theory and its role in modern cryptography. Integers k and m are *congruent modulo* an integer $n > 1$, written

$$m \equiv k \bmod n,$$

iff m and k have the same remainder on division by n. So congruence modulo n is the equivalence relation determined by the remainder-on-division-by-n function. This relation is called a *congruence* because adding or multiplying equivalent integers yields equivalent integers. That is,

Lemma 2.2. *If* $m_1 \equiv k_1 \bmod n$ *and* $m_2 \equiv k_2 \bmod n$, *then*

$$\begin{aligned} m_1 + m_2 &\equiv k_1 + k_2 \bmod n, \text{ and} \\ m_1 m_2 &\equiv k_1 k_2 \bmod n. \end{aligned}$$

We leave the proof of Lemma 2.2 as an easy exercise for the reader.

2.2 Partitions

Cutting up a set into a bunch of pieces is called *partitioning* the set. The pieces are called *blocks* of the partition (you'd think the pieces would be called the *parts* of the *part*ition, but no). More formally,

Definition 2.3. A *partition* of a set, A, is a collection, $\mathcal{A}$, of nonempty sets called the *blocks of the partition* such that

1. $A = \bigcup_{B \in \mathcal{A}} B$, and
2. if $B_1 \neq B_2$ are blocks of $\mathcal{A}$, then B_1 and B_2 are disjoint[1].

[1]Two sets are said to be disjoint when they have no elements in common, that is, their intersection is empty.

Example 2.4. We can partition the integers into four blocks according to whether their remainder on division by 4 is 0, 1, 2, or 3:

$$\begin{gathered}\{0, 4, -4, 8, -8, 12, \dots\}\\ \{1, -3, 5, -7, 9, -11, \dots\}\\ \{2, -2, 6, -6, 10, -10, \dots\}\\ \{3, -1, 7, -5, 11, -9, \dots\}\,.\end{gathered}$$

Example 2.5. We can partition the real line into blocks by cutting it at integer points. Namely, the nth block, B_n, would be $\{r \in \mathbb{R} \mid n \le r < n+1\}$. So B_n could also be described as the set of real numbers r that are $\equiv_f$ to n, where f is the *floor* function: $f(r) = \lfloor r \rfloor$, the largest integer $\le r$.

Example 2.6. We can partition the pixels in an image according to their color (so there will be somewhere between 2 and several millions blocks depending on whether the image is pure black and white or is "true color."

The relation of *being in the same block* of a partition is an equivalence relation. This is the equivalence relation defined by the function that maps each element to the block it's in. More precisely, suppose $\mathcal{A}$ partitions a set, A, and define $[a]_{\mathcal{A}}$ be the block with a in it. Note that every $a \in A$ belongs to some block by Definition 2.3.1, and there is only one such block by Definition 2.3.2, so $[a]_{\mathcal{A}}$ is unambigously defined for each element, a. So being-in-the-same-block is $\equiv_{\text{blk}}$, where $\text{blk}(a) ::= [a]_{\mathcal{A}}$.

Conversely, an equivalence relation, $\equiv_f$, given by a total function, f, on a set, A, determines a partition of A, where the block containing $a \in A$ is $\{a' \mid f(a') = f(a)\}$.

For example, there are four equivalence classes of integers under congruence mod 4. These are exactly the blocks of the partition based on remainder-by-4 of Example 2.4.

So we can extract an equivalence relation from any partition, and conversely, we can define a partition determined by any equivalence relation. In fact, it's not hard to see that if you extract an equivalence relation from a partition and then use the partition to determine an equivalence relation, you get back to the partition you started with. Likewise, if you take the parition determined by an equivalence relation and extract a partition from it, you also get back to where you started. So partitions and equivalence relations are really interchangeable ways of talking about the same thing.

To summarize: the hallmark of equivalence relations is *sameness* of some property of objects. An equivalence relation hides irrelevant differences between objects, and lets us lump together into blocks all the objects that are the "same." Conversely, equivalence is captured by the property of being in the same block.

2.3 Properties of Equivalence Relations

Equivalence relations have some obvious properties that occur so frequently they merit names:

Definition 2.7. A binary relation R on a set A is:

- *reflexive* iff for every $a \in A$,

$$a \, R \, a,$$

- *symmetric* iff for every $a, a' \in A$,
$$a\,R\,a' \quad \text{implies} \quad a'\,R\,a,$$
- *transitive* iff for every $a, b, c \in A$,
$$[a\,R\,b \text{ and } b\,R\,c] \quad \text{implies} \quad a\,R\,c.$$

Example 2.8. Let R_1 be the less-than relation, $<$, on the natural numbers. Then R_1 is transitive (since [$j < k$ and $k < l$] implies $j < l$). It is not reflexive (since $0 < 0$ is false) and not symmetric (since $0 < 1$ but not $1 < 0$).

Example 2.9. We know that if A, B, C are sets and $A \subset B$ and $B \subset C$, then $A \subset C$. That is, the proper subset relation, $\subset$, is transitive. It is not reflexive (since a set is never a *proper* subset of itself) and not symmetric (for example, the empty set is a subset of any nonempty set, but not vice versa).

Example 2.10. Let R_2 be the "implies" relation on the set of propositional formulas, that is, define $p\,R_2\,q$ iff $p \longrightarrow q$ is propositionally valid. Now R_2 is reflexive, since $p \longrightarrow p$ is valid. It is also transitive, since if $p \longrightarrow q$ and $q \longrightarrow r$ are valid, then $p \longrightarrow r$ is valid. However, it isn't symmetric, since, for example, `false` $\longrightarrow$ `true` valid, but `true` $\longrightarrow$ `false` is not.

Example 2.11. Let R_3 be the relation on sets, C, D of natural numbers such that $C\,R_3\,D$ iff $C \cap D$ is finite. Then R_3 is symmetric, but not reflexive (for example $\mathbb{N}\,R\,\mathbb{N}$ is not true).

Quick exercise: Explain why R_3 is not transitive.

Example 2.12. Let R_4 be the relation on complex numbers such that $a\,R_4\,b$ iff the distance from a to b in the complex plane is ≤ 1, that is, $|a - b| \leq 1$. Then R_4 is reflexive and symmetric, but not transitive (because $1\,R_4\,2$ and $2\,R_4\,3$, but not $1\,R_4\,3$).

Notice that the equality relation on a set A is reflexive, symmetric, and transitive. We won't prove this—it's an axiom. These properties of equality directly imply the corresponding properties for any equivalence relation:

Lemma 2.13. *Every equivalence relation is reflexive, symmetric, and transitive.*

Proof. Consider any equivalence relation, $\equiv_f$, determined by some function, f with domain A.

Since $f(a) = f(a)$, it follows trivially that $\equiv_f$ is reflexive. LIkewise, if $f(a) = f(a')$, then $f(a') = f(a)$, which implies that $\equiv_f$ is symmetric. Finally, if $f(a) = f(b)$ and $f(b) = f(c)$, than obviously $f(a) = f(c)$, which implies that $\equiv_f$ is transitive. $\square$

2.4 Equivalence by Axioms

The properties of reflexivity, symmetry, and transitivity actually provide an elegant axiomatic characterization of equivalence relations. (In fact, most authors *define* equivalence relations using these axiomas, but we think our approach makes more sense.)

Theorem 2.14. *Any relation on a set that is reflexive, symmetric and transitive is an equivalence relation on the set.*

Proof. Suppose R is a relation on a set, A, and R is reflexive, symmetric, and transitive. Define the function, f, with domain, A, by the rule

$$f(a) ::= \{a\}\,R.$$

We will prove that R is $\equiv_f$, and hence R is an equivalence relation. That is, we have to show that

$$a\,R\,b \quad \text{iff} \quad \{a\}\,R = \{b\}\,R \tag{2}$$

for all $a, b \in A$.

First we prove (2) from right to left. Namely, suppose $\{a\}\,R = \{b\}\,R$. Since R is **reflexive**, we have $b \in \{b\}\,R$. This means $b \in \{a\}\,R = \{b\}\,R$. So $a\,R\,b$ holds by definition of $\{a\}\,R$, which completes the proof from right to left.

To prove the converse, suppose

$$a\,R\,b. \tag{3}$$

We'll first prove that

$$\{b\}\,R \subseteq \{a\}\,R. \tag{4}$$

To do this, let c be an element of $\{b\}\,R$; we must show that $c \in \{a\}\,R$. But by definition of $\{b\}\,R$, we know that

$$b\,R\,c. \tag{5}$$

But (3) and (5) together imply $a\,R\,c$ because R is **transitive**. So $c \in \{a\}\,R$ by definition of $\{a\}\,R$. This proves (4).

Finally, (3) implies $b\,R\,a$, because R is **symmetric**. So the same argument used to prove (4), we can conclude that

$$\{a\}\,R \subseteq \{b\}\,R.$$

But this together with (4) implies that $\{a\}\,R = \{b\}\,R$, completing the proof of (2) from left to right. $\square$

We have highlighted where each of the three properties of equivalence relations were used in this proof. Verifying that a proof uses all available assumptions is a good "sanity check"—if one of the assumed properties was not used in the proof, then you have either made a mistake or proven a stronger theorem than you thought. For example, if you didn't use reflexivity anywhere in the proof of Theorem 2.14, you would have proved that any symmetric, transitive relation is an equivalence relation, which is false. So a proof that failed to use reflexivity must be mistaken.

These axioms are helpful in at least a couple of ways. First, there are situations where it's hard to find a function, f, that characterizes an equivalence relation. Theorem 2.14 lets us show that the relation is an equivalence by verifying that the relation is reflexive, symmetric, and transitive.

Problem 1. Define two positions of the pieces in the game of chess to be *mutually reachable* if it is possible to start at either one and get to the other one by a sequence of legal chess moves. We don't see any useful way to describe a function, f, such that mutual reachability equals $\equiv_f$. Prove that mutual reachability is an equivalence relation anyway.

Second, to prove that a relation is *not* an equivalence just from the definition, we would have to show that the relation is not equal to $\equiv_f$ for any possible function, f. Off hand, this requires a daunting analysis in order to rule out all possible functions. But Theorem 2.14 implies we can always show that a relation is not an equivalence simply by finding two or three elements of the domain where the axioms fail to hold.

3 Partial Orders

Partial orders are another class of binary relations that are particularly important in Computer Science, with applications that include task scheduling, database concurrency control, and proving that computations terminate.

A general example of a partial order is the subset relation, $\subset$, on sets. In fact, we will *define* partial orders via the subset relation in much the same way we defined equivalence relations. Namely, for any element, a, we think of a function, g, such that $g(a)$ is the *set* of properties that a has. Then we relate different elements according to how their properties compare. All partial orders will arise in this way.

3.1 Partial Order by Function

For partial orders we'll often use the symbols $\prec$ or $\preceq$ because they resemble the symbols used for subset and less-or-equal, which are the most common partial orders. (General relations are usually denoted by a letter like R instead of a cryptic squiggly symbol, so $\preceq$ is kind of like Prince.)

Definition 3.1. Given any total function, g, from a set, A, to a collection of sets, define the binary relation $\prec_g$ on A by the rule:

$$a \prec_g b \quad \text{iff} \quad g(a) \subset g(b) \tag{6}$$

for $a, b \in A$. A binary relation, R, on a set, A, is a *partial order* iff there is a g such that R agrees with $\prec_g$ for every pair of distinct elements. That is,

$$a \, R \, b \quad \text{iff} \quad a \prec_g b \tag{7}$$

for all $a \neq b \in A$.

An immediate consequence of Definition 3.1 is that the subset relation itself is a partial order. Specifically, if A is any collection of sets, then the proper subset relation, $\subset$, is a partial order on A. To prove this, we let I_A be the identity function on A. Then $\prec_{I_A}$ is the same relation as $\subset$, and so trivially satisfies the condition (6) on R. The contained-in-or-equal subset relation, $\subseteq$, is also a partial order, since it agrees with $\prec_{I_A}$ for $a \neq b$.

The most familiar examples of partial orders are "less than" relations, for example, the relation, $<$, on real numbers. To see that $<$ is indeed a partial order, just define $h(r) ::= \{q \in \mathbb{Q} \mid q < r\}$. Since there is a rational number between any two real numbers (to see why, think about where their decimal expansions first differ), it follows that $<$ is simply $\prec_h$. Likewise, the relation, $\leq$, is a partial order because it agrees with $\prec_h$ for all pairs of distinct real numbers.

Our general definition of partial order leaves unspecified whether elements are related to themselves.

Definition 3.2. A partial order is called *weak* iff it is reflexive.

So, for example, the relation $\leq$ on the real numbers, and the relation $\subseteq$ on sets, are weak partial orders.

Definition 3.3. A binary relation, R, on a set A, is *irreflexive* iff for all $a \in A$ it is *not* true that $a \, R \, a$.

A partial order is *strict* iff it is irreflexive.

The $<$-relation on the reals and the proper subset relation, $\subset$, are strict partial orders. In general, a partial order may be neither weak nor strict; this happens when some elements are related to themselves and others are not.

Two more examples of partial orders are worth mentioning:

Example 3.4. Let A be some family of sets and define aRb iff $a \supset b$. Then R is a strict partial order.

Proof. Define $p(a) ::= \overline{a}$, where $\overline{a}$ is the complement of a, and note that

$$[a\,R\,b] \quad \text{iff} \quad [a \supset b] \quad \text{iff} \quad [\overline{a} \subset \overline{b}] \quad \text{iff} \quad [p(a) \subset p(b)].$$

So R equals $\prec_p$ and so is a partial order. It is strict since no set is a proper subset of itself. □

For integers, m, n we write $m \mid n$ to mean that m *divides* n, namely, there is an integer, k, such that $n = km$.

Example 3.5. The divides relation is a weak partial order on the natural numbers.

Proof. Let $v(n) ::=$ the set of natural numbers that divide n. Then divides is $\prec_v$, and so is a partial order. Since $m \mid m$, it is a weak partial order. □

3.2 Total Orders

The familiar order relations on numbers have an important additional property: given any two numbers, one will be bigger than the other. Partial orders with this property are said to be *total orders*[2]:

Definition 3.6. Let R be a binary relation on a set, A, and let a, b be elements of A. Then a and b are *comparable* with respect to R iff ($a\,R\,b$ or $b\,R\,a$). A partial order under which every two distinct elements are comparable is called a *total order*.

So $<$ and $\leq$ are total orders on $\mathbb{R}$. On the other hand, the subset relation is generally *not* total: any two distinct finite sets of the same size will be incomparable under $\subseteq$.

3.3 Properties of Partial Orders

We've already observed that the subset relation is transitive, which implies that any relation, $\prec_g$, is transitive. So by Definiton 7:

Lemma 3.7. *Every partial order is transitive.*

One additional property of the subset relation is enough to completely characterize partial orders.

Definition 3.8. A binary relation, R, on a set, A, is *antisymmetric* if

$$a\,R\,b \quad \text{implies} \quad \neg(b\,R\,a)$$

for all $a \neq b \in A$.

[2]"Total" is an overloaded term when talking about partial orders: being a total order is a much stronger condition that being a partial order that is a total relation. For example, any weak partial order such as $\subseteq$ is a total relation.

Lemma 3.9. *Every partial order is antisymmetric.*

Proof. Suppose R is a partial order on A. So there is a set-valued total function, g, with domain A and R agrees with $\prec_g$ on pairs of distinct elements in A.

We want to show that aRb and bRa cannot both hold for elements $a \neq b$. But if they did both hold, then (6) would imply that $g(a)$ and $g(b)$ are proper subsets of each other, which is impossible. $\square$

3.4 Partial Orders by Axioms

The properties of transitivity and antisymmetry provide an axiomatic characterization of partial orders:

Theorem 3.10. *A binary relation is a partial order iff it is transitive and antisymmetric.*

Proof. Let R be a binary relation on a set A. Then the preceding two Lemmas imply the left to right direction of Theorem 3.10.

To prove the Theorem in the right to left direction, assume R is transitive and antisymmetric. Define

$$g(a) ::= R\{a\} \cup \{a\}\,. \tag{8}$$

We claim that R is a partial because it agrees with $\prec_g$ on distinct elements of A. That is, if $a \neq b \in A$, then

$$a\,R\,b \quad \text{iff} \quad g(a) \subset g(b). \tag{9}$$

To prove (9) from right to left, note that

$$\begin{aligned}
g(a) \subset g(b) \quad &\text{implies} \quad a \in g(b) && \text{because } a \in g(a) \text{ by (8)},\\
&\text{implies} \quad a \in R\{b\} \cup \{b\} && \text{by def. of } g(b),\\
&\text{implies} \quad a \in R\{b\} && \text{since } a \neq b,\\
&\text{implies} \quad a\,R\,b && \text{by def of } R\{b\}.
\end{aligned}$$

The proof of (9) from left to right also follows routinely from the definitions by a somewhat longer, but not specially informative argument, which we omit.

$\square$

In the literature, partial orders are usually defined axiomatically as in Theorem 3.10, and then the possibility of representing partial orders as $\prec_g$ is proved as a theorem. Since the two characterizations imply each other, it is a matter of taste which one to use as the definition.

Strict partial orders have an even simpler axiomatic characterization.

Theorem 3.11. *A binary relation is a strict partial order iff it is transitive and irreflexive.*

Problem 2. Prove Theorem 3.11. *Hint:* Show that transitivity and irreflexivity imply antisymmetry.

For weak partial orders, we often write an ordering-style symbol like $\preceq$ instead of a letter symbol like R. Likewise, we generally use $\prec$ to indicate a strict partial order. We also write $b \succeq a$ to mean $a \preceq b$ and $b \succ a$ to mean $a \prec b$.

3.5 Products and Restrictions of Relations

Product and restriction are two ways of constructing new relations from old ones that will be useful.

3.5.1 Products

The product, $R_1 \times R_2$, of relations R_1 and R_2 is defined to be the relation with

$$\begin{aligned} \text{domain}\,(R_1 \times R_2) &::= \text{domain}\,(R_1) \times \text{domain}\,(R_2)\,, \\ \text{codomain}\,(R_1 \times R_2) &::= \text{codomain}\,(R_1) \times \text{codomain}\,(R_2)\,, \\ (a_1, a_2)\,(R_1 \times R_2)\,(b_1, b_2) &\quad \text{iff} \quad [a_1\, R_1\, b_1 \text{ and } a_2\, R_2\, b_2]. \end{aligned}$$

Example 3.12. Define a relation, Y, on age-height pairs of being younger *and* shorter. This is the relation on the set of pairs (y, h) where y is a natural number ≤ 2400 which we interpret as an age in months, and h is a natural number ≤ 120 describing height in inches. We define Y by the rule

$$(y_1, h_1)\, Y\, (y_2, h_2) \quad \text{iff} \quad y_1 \leq y_2 \wedge h_1 \leq h_2.$$

That is, Y is the product of the $\leq$-relation on ages and the $\leq$-relation on heights.

Products preserve several of the relational properties we have considered. Namely, it's not hard to verify that if R_1 and R_2 are both transitive, then so is $R_1 \times R_2$. The same holds for symmetry, reflexivity, and antisymmetry. This implies that if R_1 and R_2 are both partial orders, then so is $R_1 \times R_2$. Likewise for being an equivalence relation.

Quick Exercise: Verify that if either of R_1 or R_2 is irreflexive, then so is $R_1 \times R_2$.

On the other hand, the property of being a total order is not preserved. For example, the age-height relation Y is the product of two total orders, but it is not total. For example, the age 240 months, height 68 inches pair, (240,68), and the pair (228,72) are incomparable under Y.

3.5.2 Restrictions

We usually think of a single "less-than" relation, $<$, on real numbers, or rational numbers, or integers. Technically, these a different relations because they have different domains and graphs, but there is an obvious connection: the $<$ order on, say, the rationals is gotten by restricting the real order to the subset of rationals.

Definition 3.13. Let R be a relation on a set, A, and let B be a subset of A. The *restriction* of R to B is the relation on B whose graph is $\text{graph}\,(R) \cap (B \times B)$.

Restrictions preserve many relational properties. For example, restriction preserves transitivity, that is, if R is transitive, then so is any restriction of R. Restriction also preserves symmetry, antisymmetry, asymmetry, reflexivity, and irreflexivity. This implies that the restriction of an equivalence relation is an equivalence relation, and the restriction of a partial order is a partial order.

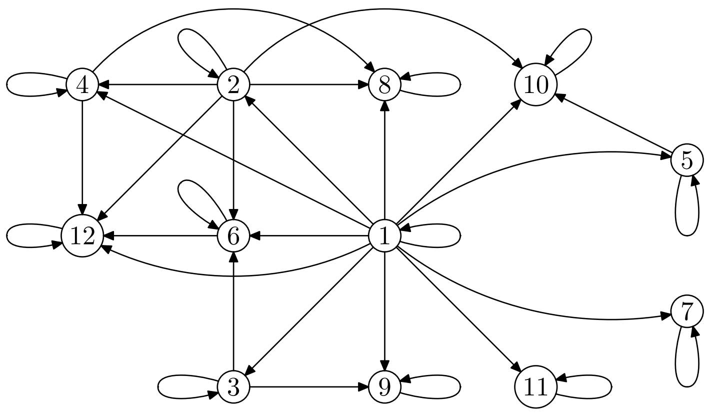
But restriction doesn't preserve all the relational properties we've considered. For example, being a surjective relation is not preserved by restriction, nor is being a total relation.

We'll leave the proofs of these claims to the reader; they're all easy.

4 Digraphs

A *directed graph* (*digraph* for short) is formally the same as a binary relation on a set, A, but we picture the digraph geometrically by representing elements of A as points on the plane, with an arrow from the point for a to the point for b exactly when $a \, R \, b$. The elements of A are referred to as the *vertices* of the digraph.

Example 4.1. The divisibility relation on $\{1, 2, \ldots, 12\}$ is represented by the digraph:



4.1 Paths in Digraphs

Definition 4.2. A *path in a digraph*, R, is a sequence of vertices $a_0, \ldots, a_k$ with $k \geq 0$ such that $a_i \, R \, a_{i+1}$ for every $0 \leq i < k$. The path is said to *start* at a_0, to *end* at a_k, and the *length* of the path is defined to be k.

Pictured with points and arrows, a path $a_0, \ldots, a_k$ looks like a line that starts at the point a_0 and follows arrows between successive points on the path to end at a_k. Note that a single point counts as a length zero path (this is just for convenience).

Many of the relational properties have geometric descriptions in terms of digraphs. For example:

Reflexivity: All vertices have self-loops (a *self-loop* at a vertex is an arrow going from the vertex back to itself).

Irreflexivity: No vertices have self-loops.

Symmetry: All edges are bidirectional.

Transitivity: Short-circuits—for any path through the graph, there is an arrow from the first vertex to the last vertex on the path.

We can define some new relations based on paths. Let R be a digraph with vertices, A. Define relations R^* and R^+ on A by the conditions that

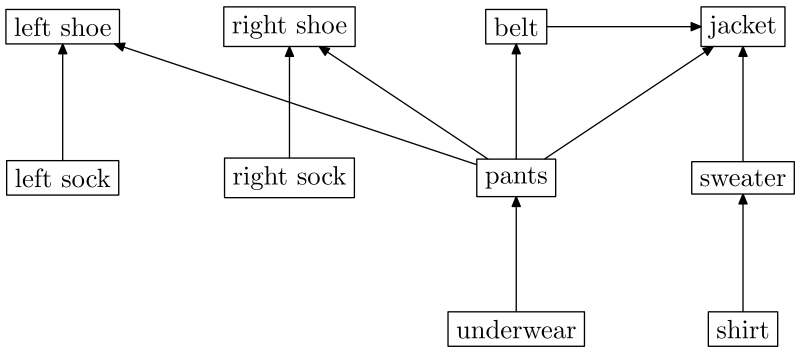
$$\begin{aligned} a \, R^* \, b \quad &::= \quad \text{there is a path in } R \text{ from } a \text{ to } b, \\ a \, R^+ \, b \quad &::= \quad \text{there is a positive length path in } R \text{ from } a \text{ to } b. \end{aligned}$$

R^* is called the *path relation* of R. It follows from the definition of path that R^* is transitive. It is also reflexive (because of the length-zero paths) and it contains the graph of R (because of the length-one paths). R^+ is called the *positive-length path relation*; it also contains graph (R) and is transitive.

4.2 Directed Acyclic Graphs

Scheduling problems are a common source of partial orders: there is a set, A, of tasks and a set of constraints specifying that starting a certain task depends on other tasks being completed beforehand. We represent the task by vertices and a constraint that task a must finish before task b can start by an arrow from a to b.

Example 4.3. Here is a graph that describes the order in which you could put on clothes. The tasks are the clothes to be put on, and the edges say what should be put on before what.



This "depends on" graph imposes a partial ordering on tasks. But what if we add a relation edge from belt to underwear? In that case the dependency graph stops making sense: there is no way to get dressed! What goes wrong is that the added edge creates a "cyclic" dependency.

Definition 4.4. A *cycle* is a positive length path in a digraph that begins and ends at the same vertex. A *directed acyclic graph (DAG)* is a directed graph with no cycles.

So a task graph had better be a DAG for its tasks to be doable in an order that respects task dependencies.

We use DAG's as an economical way to represent the dependency relation. Usually a task-graph DAG itself is not a transitive relation because it includes only the edges showing "direct" dependencies. Rather, the *dependency relation* we care about is defined by the positive length path relation, R^+, in the task graph. The dependency relation will always be a partial order:

Lemma 4.5. *If D is a DAG, then D^+ is a strict partial order.*

Proof. We know that D^+ is transitive. Also, a positive length path from a vertex to itself would be a cycle, so there are no such paths. This means D^+ is irreflexive, and so by Theorem 3.11, it is a strict partial order. □

4.3 Topological Sorting

In a DAG for a partial order, incomparable elements appear as vertices with no path between them in either direction. So in the partial order on clothes from Example 4.3 “left shoe” and “right shoe” are incomparable. If the order is total, there are no incomparable elements, and the order can be represented by a DAG looks like a line:

$$\begin{tikzcd}
\circ \arrow[d] \\
\circ \arrow[d] \\
\vdots \arrow[d] \\
\circ
\end{tikzcd}$$

When we have a partial order of tasks to be performed, it can be useful to have an order in which to perform all the tasks, one at a time, while respecting the dependency constraints. This amounts to finding a total order that is consistent with the partial order. This task of finding a total ordering that is consistent with a partial order is known as *topological sorting*.

Definition 4.6. A *topological sort* of a partial order, $\prec$, on a set, A, is a total ordering, $\sqsubset$, on A such that

$$a \prec b \quad \text{implies} \quad a \sqsubset b.$$

For example,

$$\text{shirt} \sqsubset \text{sweater} \sqsubset \text{underwear} \sqsubset \text{leftsock} \sqsubset \text{rightsock} \sqsubset \text{pants} \sqsubset \text{leftshoe} \sqsubset \text{rightshoe} \sqsubset \text{belt} \sqsubset \text{jacket},$$

is one topological sort of the partial order of dressing tasks given by the DAG of Example 4.3; there are several other possible sorts as well.

Topological sorts for finite DAG’s are easy to construct by starting from *minimal* elements:

Definition 4.7. Let $\preceq$ be a partial order on a set, A, and let a be an element of A. Then a is *minimal* iff no other element is $\preceq a$. Similarly, a is *maximal* iff no other element is $\succeq a$.

In a total order, there can only be one minimal element, but in general there can be more than one minimal element in a partial order. There are four in the clothes example: leftsock, rightsock, underwear, and shirt. To construct a total ordering for getting dressed, we pick one of these minimal elements, say shirt. Next we pick a minimal element among the remaining ones. For example, once we have removed shirt, sweater becomes minimal. We continue in this way removing successive minimal elements until all elements have been picked. The sequence of elements in the order they were picked will be a topological sort. This is how the topological sort above for getting dressed was contructed.

For this method of topological sorting to work, we need to be sure there is always a minimal element. (An infinite partially ordered set might have no minimal element: consider $\leq$ on the $\mathbb{Z}$.)

Lemma 4.8. *Every partial order on a nonempty finite set has a minimal element.*

Proof. Let R be a partial order on a set, A. For any element, $a \in A$, let $g(a)$ be the set of elements "less than or equal to a", that is,

$$g(a) ::= R\{a\} \cup \{a\}.$$

Now if $b\,R\,a$, then transitivity of R implies that $g(b) \subseteq g(a)$. Also, if $b\,R\,a$ and $b \neq a$, then $a \notin g(b)$ since R is antisymmetric, and so $g(b) \subset g(a)$. So if a is not minimal, then there is some b such that $g(b) \subset g(a)$. If A is finite, this implies that $|g(b)| < |g(a)|$.

So if A is finite, the Well Ordering Principle implies that there must be an a_0 such that $g(a_0)$ has minimum size. So no $g(b)$ can be smaller than $g(a_0)$, which means a_0 must be minimal. □

Theorem 4.9. *Every partial order on a finite set has a topological sort.*

Proof. We prove Theorem 4.9 by induction on n with hypothesis

$$P(n) ::= [\text{any partial order on a set with } n \text{ elements has a topological sort}].$$

Base case $n = 1$: a topological sort of a set with one element is simply that element.

Inductive step: Assume $P(n)$. Consider a partial order, $\preceq$ on a set, A, with $n + 1$ elements. By Lemma 4.8, A must have a minimal element, a_0. Now the restriction of $\preceq$ to the set $A - \{a_0\}$ is also a partial order. So by the inductive hypothesis, $A - \{a_0\}$ has a topological sort $\sqsubseteq_n$. Now define $\sqsubseteq$ on A by the rule that $a \sqsubseteq b$ iff $[a \sqsubseteq_n b$ or $a = a_0]$. It's now easy to check that $\sqsubseteq$ is the required topological sort of A. This proves $P(n + 1)$, completing the proof by induction. □

There are many other ways of contructing topological sorts. In fact, the domain of the partial order need not be finite: we won't prove it, but *all* partial orders, even infinite ones, have topological sorts.

4.4 Parallel Task Scheduling

For the partial order of dependencies among task, topological sorting provides a way to execute tasks sequentially without violating the dependencies. But what if we have the ability to execute more than one task at the same time? For example, say tasks are programs, the partial order indicates data dependence, and we have a parallel machine with lots of processors instead of a sequential machine with only one. How should we schedule the tasks? Our goal should be to minimize the total *time* to complete all the tasks. For simplicity, let's say all the tasks take the same amount of time and all the processors are identical.

So, given a finite partially ordered set of tasks, how long does it take to do them all, in an optimal parallel schedule? We can also use partial order concepts to analyze this problem.

In the clothes example, we could do all the minimal elements first (leftsock, rightsock, underwear, shirt), remove them and repeat. We'd need lots of hands, or maybe dressing servants. We can do pants and sweater next, and then leftshoe, rightshoe, and belt, and finally jacket.

We can't do any better, because the sequence underwear, pants, belt, jacket must be done in that order. A set of asks that must be done in sequence like this is called a chain.

Definition 4.10. A *chain* in a partial order is a set of elements such that any two elements in the set are comparable.

An alternative definition is that a chain in a partial order is a set, C, domain elements such that the restriction of the partial order to C is a total order.

Note that vertices on any path in the DAG of the partial order is a chain. In general, a chain only contains vertices on a path but may skip some vertices on it. Clearly, the parallel time must be at least the size of any chain. For if we used less time, then two tasks in the chain would have to be done at the same time, violating the dependency constraints.

A largest chain is also known as a *critical path*. So we need at least t steps, where t is the size[3] of a largest chain. Fortunately, it is always possible to use only t parallel steps:

Theorem 4.11. *Let R be strict partial order on a set, A. If the longest chain in A is of size t, then there is a partition of A into t blocks, $B_1, B_2, \ldots, B_t$, such that for each block, B_i, all tasks that have to precede tasks in B_i are in smaller-numbered groups:*

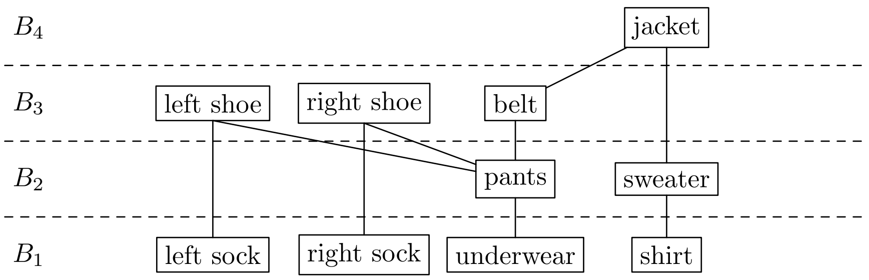
$$RB_1 = \emptyset, \text{ and} \tag{10}$$

$$RB_i \subseteq B_1 \cup B_2 \cup \cdots \cup B_{i-1}, \tag{11}$$

for $1 < i \leq t$.

Corollary 4.12. *For R and t as above, it is possible to schedule all tasks in t steps.*

Proof. Schedule all the elements of B_i at time i. This satisfies the dependency requirements, because all the tasks that any task depends on are scheduled at preceding times. □



Corollary 4.13. *Parallel time = Size of largest chain.*

So it remains to prove Theorem 4.11:

Proof. Construct the sets B_i as follows:

$$B_i ::= \{a \in A \mid \text{the largest chain ending in } a \text{ is of size } i\}.$$

This gives just t sets, because the largest chain is of size t. Also, each $a \in A$ belongs to exactly one B_i. To complete the proof, notice that if $a \in B_1$, then a must be minimal, and since R is strict we have $RB_1 = \emptyset$ proving (10).

Now suppose $1 < i \leq t$, and assume for the sake of contradiction that (11) does not hold. That is, there is an $a \in B_i$ and $b \in A$ such that $b \, R \, a$, and $b \notin B_1 \cup B_2 \cup \cdots \cup B_{i-1}$. Then by definition of the B_j's, there is a chain of size $> i - 1$ ending at b. Also, since R is strict, a is not in the chain ending at b. So we can add a to the end of the chain to obtain a chain of size $> i$ ending in a, contradicting the fact that that $a \in B_i$. □

[3]Picky point: the *length* of a chain $a_0 \prec a_1 \cdots \prec a_k$ is k, corresponding to the number of arrows it traverses. The *size* of the chain is the number of elements in it, namely, $k + 1$.

So with an unlimited number of processors, the time to complete all the tasks is the size of the largest chain. It turns out that this theorem is good for more than parallel scheduling. It is usually stated as follows.

Definition 4.14. An *antichain* in a partial order is a set of elements such that any two elements in the set are incomparable.

Corollary 4.15. *If the largest chain in a partial order is of size, t, then the domain can be partitioned into t antichains.*

Proof. Let the antichains be the sets B_i defined as in the proof of Theorem 4.11.

We should verify that each B_i is an antichain, namely, if a, b are distinct elements of B_i, then they are incomparable. But suppose to the contrary that there exist two elements $a, b \in B_i$ such that a and b are comparable, say $a\,R\,b$. Then, as in the proof of Theorem 4.11, by adding b at the end of the chain of size i ending at a, we obtain a chain of size $i + 1$ ending at b, contradicting the assumption that $b \in B_i$. $\square$

4.5 Dilworth's Lemma

We can use the Corollary 4.15 to prove a famous result[4] about partially ordered sets:

Lemma 4.16 (Dilworth). *For all t, every partially ordered set with n elements must have either a chain of size greater than t or an antichain of size at least n/t.*

Proof. Assume there is no chain of size greater than t, that is, the largest chain is of size $\leq t$. Then by Corollary 4.15, the n elements can be partitioned into at most t antichains. Let ℓ be the size of the largest antichain. Since every element belongs to exactly one antichain, and there are at most t antichains, there can't be more than ℓt elements, namely, $\ell t \geq n$. So there is an antichain with at least $\ell \geq n/t$ elements. $\square$

Corollary 4.17. *Every partially ordered set with n elements has a chain of size greater than $\sqrt{n}$ or an antichain of size at least $\sqrt{n}$.*

Proof. Set $t = \sqrt{n}$ in Lemma 4.16. $\square$

Example 4.18. In the dressing partially ordered set, $n = 10$. Try $t = 3$. Has a chain of size 4. Try $t = 4$. Has no chain of size 5, but has an antichain of size $4 \geq 10/4$.

Example 4.19. Suppose we have a class of 101 students. Then using the product partial order, Y, from Example 3.12, we can apply Dilworth's Lemma to conclude that there is a chain of 11 students who get taller as they get older, or an antichain of 11 students who get taller as they get younger, which makes for an amusing in-class demo.

As a curious consequence of Corollary 4.17, we have:

Corollary 4.20. *In any sequence of n different numbers, there is either an increasing subsequence of length greater than $\sqrt{n}$ or a decreasing subsequence of length at least $\sqrt{n}$.*

[4] Lemma 4.16 also follows from a more general result known as Dilworth's Theorem which we will not discuss.

Example 4.21. The sequence

$$\langle 6, 4, 7, 9, 1, 2, 5, 3, 8 \rangle$$

has the decreasing sequence $\langle 6, 4, 1 \rangle$ and the increasing sequence $\langle 1, 2, 3, 8 \rangle$.

Proof. We can prove Corollary 4.20 using Dilworth's Lemma; the trick is to define the appropriate partially ordered set. Suppose the given sequence is

$$a_1 < a_2 < \cdots < a_n.$$

Let the domain of the partial order be the set of pairs (i, a_i) for $1 \le i \le n$, and define

$$(i, a_i) \prec (j, a_j) \quad \text{iff} \quad i < j \land a_i < a_j.$$

So $\prec$ is a strict partial order because it is a restriction of the product of the $<$ relations on $\{1, \dots, n\}$ and $\{a_1, \dots, a_n\}$

Now given a $\prec$-chain, if we arrange the elements so that their first coordinates are in increasing order, their second coordinates must also be in increasing order. That is, if we have

$$(i_1, a_{i_1}) \prec (i_2, a_{i_2}) \prec \cdots \prec (i_k, a_{i_k}),$$

then the i_j's and the a_{i_j}'s both increase from left to right. This means that a *chain* corresponds to an *increasing* subsequence.

But what does an antichain correspond to? Well, suppose $i < j$ and (i, a_i) and (j, a_j) are $\prec$-incomparable; then by definition, we must have $a_i > a_j$. So given an antichain, if we arrange the elements so that their first coordinates are in increasing order, their second coordinates are in decreasing order. That is, an *antichain*, with its elements sorted on their first coordinates, corresponds to a *decreasing* subsequence.

Corollary 4.20 now follows immediately from Dilworth's Lemma. □

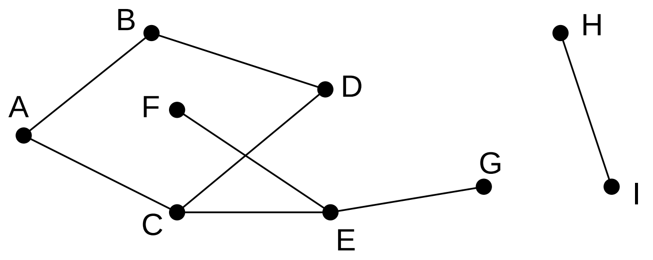
Quick Exercise: What is the size of the longest chain that is guaranteed to exist in any partially ordered set of n elements? What about the largest antichain?

Problem 3. Describe a sequence consisting of the integers from 1 to 10,000 in some order so that there is no increasing or decreasing subsequence of size 101.

Not So Quick Exercise: Devise an efficient procedure for finding the longest increasing and the longest decreasing subsequence in any given sequence of integers. (There is a nice one.)

5 Undirected Graphs

Informally, an *undirected graph*, or *ugraph* for short, is a bunch of dots connected by lines. Here is an example of a graph:



Sadly, this definition is not precise enough for mathematical discussion.

Definition 5.1. A ugraph, G, consists of a set, V, called the *vertices* of G, and a collection, E, of two element subsets of V. The elements of E are called the *edges* of G.

The vertices correspond to the dots in the picture, and the edges correspond to the lines. Thus, the dots-and-lines diagram above is a pictorial representation of the ugraph where:

$$\begin{aligned} V &= \{A, B, C, D, E, F, G, H, I\} \\ E &= \{\{A, B\}, \{A, C\}, \{B, D\}, \{C, D\}, \{C, E\}, \{E, F\}, \{E, G\}, \{H, I\}\}. \end{aligned}$$

It will be helpful to use the notation A—B for the edge $\{A, B\}$. Note that A—B and B—A are different descriptions of the same edge, since sets are unordered.

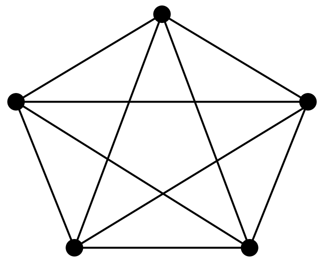
Two vertices in a graph are said to be *adjacent* if they are joined by an edge, and an edge is said to be *incident* to the vertices it joins. The number of edges incident to a vertex is called the *degree* of the vertex. For example, in the graph above, A is adjacent to B and B is adjacent to D, and the edge A—C is incident to vertices A and C. Vertex H has degree 1, D has degree 2, and E has degree 3.

Ugraphs are essentially the same as symmetric relations or digraphs with a reverse arrow for every arrow. They have surprisingly many applications, for example, in solving routing, schedule conflict, and molecular structure problems.

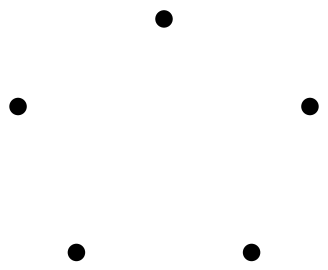
In the remainder of this section, we'll refer to ugraphs simply as *graphs*.

5.1 Some Common Graphs

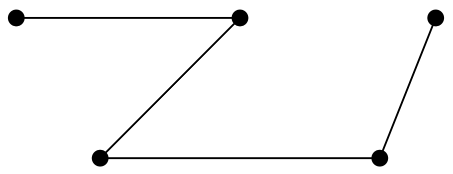
Some graphs come up so frequently that they have names. The *complete graph* on n vertices, also called K_n, has an edge between every two vertices. Here is K_5:



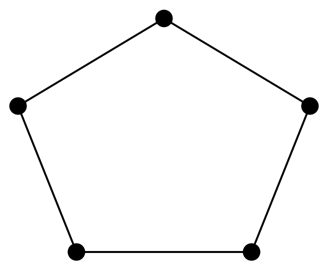
The *empty graph* has no edges at all. Here is the empty graph on 5 vertices:



Another 5 vertex graph is L_4, the *line graph* of length four:

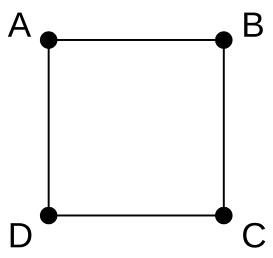


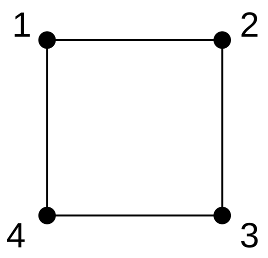
And here is C_5, a *simple cycle* with 5 vertices:



5.2 Isomorphism

Two graphs that look the same might actually be different in a formal sense. For example, the two graphs below are both simple cycles with 4 vertices:





But one graph has vertex set $\{A, B, C, D\}$ while the other has vertex set $\{1, 2, 3, 4\}$. If so, then the graphs are different mathematical objects, strictly speaking. But this is a frustrating distinction; the graphs *look the same*!

Fortunately, we can neatly capture the idea of "looks the same" and use that as our main notion of equivalence between graphs. Graphs G_1 and G_2 are *isomorphic* if there exists a one-to-one correspondence between vertices in G_1 and vertices in G_2 such that there is an edge between two vertices in G_1 if and only if there is an edge between the two corresponding vertices in G_2. For example, take the following correspondence between vertices in the two graphs above:

A corresponds to 1 $\qquad$ B corresponds to 2
D corresponds to 4 $\qquad$ C corresponds to 3.

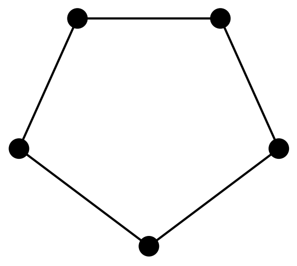
Now there is an edge between two vertices in the graph on the left if and only if there is an edge between the two corresponding vertices in the graph on the right. Therefore, the two graphs are isomorphic. The correspondence itself is called an *isomorphism*.

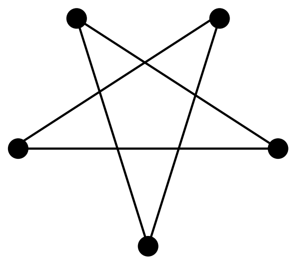
In more formal terms, if G_1 is a graph with vertices, V_1, and edges, E_1, and likewise for G_2, then G_1 is *isomorphic* to G_2 iff there exists a **bijective** function $f : V_1 \to V_2$ such that for every pair of vertices $u, v \in V_1$:

$$u\text{—}v \in E_1 \quad \text{iff} \quad f(u)\text{—}f(v) \in E_2.$$

The function f that defines the correspondence between vertices is called an *isomorphism*.

Two isomorphic graphs may be drawn to look quite different. For example, here are two different ways of drawing C_5:





Isomorphism captures all the connection properties of a graph, abstracting out what the vertices are called, what they are made out of, or where they appear in a drawing of the graph. So a property like "having three vertices of degree 4" is preserved under isomorphism, while "having a vertex that is an integer" is not preserved. In particular, if one graph has three vertices of degree 4 and another does not, they can't be isomorphic. Similarly, if one graph has an edge that is incident to degree 8 vertex and a degree 3 vertex, then any isomorphic graph must also have such an edge.

Looking for properties like these can make it easy to determine that two graphs are not isomorphic, or to actually find an isomorphism between them, if there is one. In practice, this frequently makes the problem of deciding if two graphs are isomorphic fairly easy. However, no one has yet found a *general* procedure for determining whether two graphs are isomorphic which is *guaranteed* to run much faster than an exhaustive (and exhausting) search through all possible bijections between their sets of vertices.

Having an efficient isomorphism finding/testing procedure would, for example, make it easy to search for a particular molecule in a database given the molecular bonds. On other hand, knowing there was no such efficient procedure would also be valuable: it would justify the security of a secure, reusable personal identification protocol that would be great for internet commerce. We'll explain this further in lecture on Friday.