10 Directed graphs & Partial Orders

Directed graphs, called *digraphs* for short, provide a handy way to represent how things are connected together and how to get from one thing to another by following those connections. They are usually pictured as a bunch of dots or circles with arrows between some of the dots, as in Figure 10.1. The dots are called *nodes* or *vertices* and the lines are called *directed edges* or *arrows*; the digraph in Figure 10.1 has 4 nodes and 6 directed edges.

Digraphs appear everywhere in computer science. For example, the digraph in Figure 10.2 represents a communication net, a topic we'll explore in depth in Chapter 11. Figure 10.2 has three "in" nodes (pictured as little squares) representing locations where packets may arrive at the net, the three "out" nodes representing destination locations for packets, and the remaining six nodes (pictured with little circles) represent switches. The 16 edges indicate paths that packets can take through the router.

Another place digraphs emerge in computer science is in the hyperlink structure of the World Wide Web. Letting the vertices $x_1, \ldots, x_n$ correspond to web pages, and using arrows to indicate when one page has a hyperlink to another, results in a digraph like the one in Figure 10.3—although the graph of the real World Wide Web would have n be a number in the billions and probably even the trillions. At first glance, this graph wouldn't seem to be very interesting. But in 1995, two students at Stanford, Larry Page and Sergey Brin, ultimately became multibillionaires from the realization of how useful the structure of this graph could be in building a search engine. So pay attention to graph theory, and who knows what might happen!

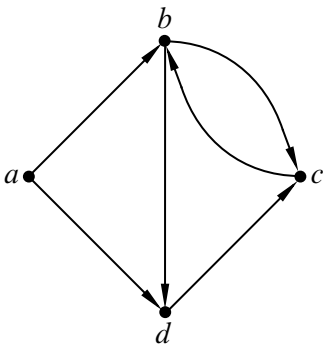


Figure 10.1 A 4-node directed graph with 6 edges.

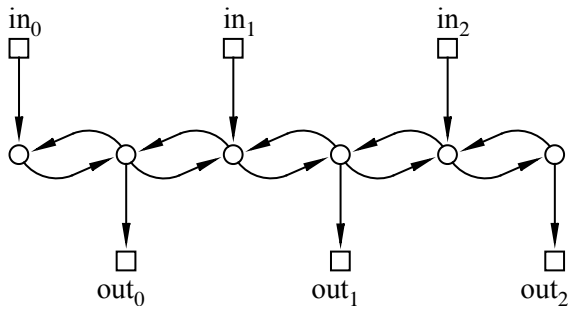


Figure 10.2 A 6-switch packet routing digraph.

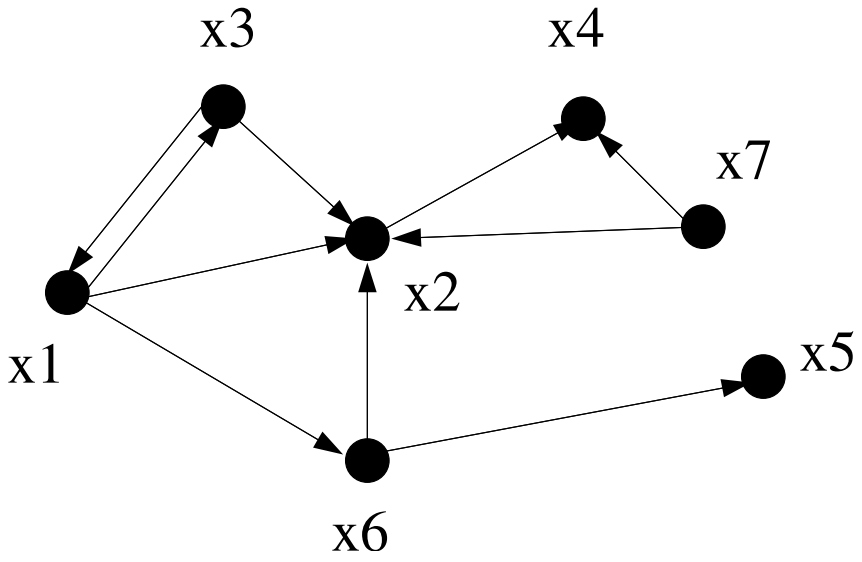


Figure 10.3 Links among Web Pages.

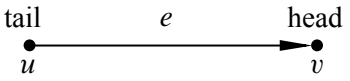


Figure 10.4 A directed edge $e = \langle u \to v \rangle$. The edge e starts at the tail vertex u and ends at the head vertex v.

Definition 10.0.1. A *directed graph* G consists of a nonempty set $V(G)$, called the *vertices* of G, and a set $E(G)$, called the *edges* of G. An element of $V(G)$ is called a *vertex*. A vertex is also called a *node*; the words "vertex" and "node" are used interchangeably. An element of $E(G)$ is called a *directed edge*. A directed edge is also called an "arrow" or simply an "edge." A directed edge *starts* at some vertex u called the *tail* of the edge, and *ends* at some vertex v called the *head* of the edge, as in Figure 10.4. Such an edge can be represented by the ordered pair (u, v). The notation $\langle u \to v \rangle$ denotes this edge.

There is nothing new in Definition 10.0.1 except for a lot of vocabulary. Formally, a digraph G is the same as a binary relation on the set, $V = V(G)$—that is, a digraph is just a binary relation whose domain and codomain are the same set V. In fact, we've already referred to the arrows in a relation G as the "graph" of G. For example, the divisibility relation on the integers in the interval [1..12] could be pictured by the digraph in Figure 10.5.

10.1 Vertex Degrees

The *in-degree* of a vertex in a digraph is the number of arrows coming into it, and similarly its *out-degree* is the number of arrows out of it. More precisely,

Definition 10.1.1. If G is a digraph and $v \in V(G)$, then

$$\text{indeg}(v) ::= |\{e \in E(G) \mid \text{head}(e) = v\}|$$
$$\text{outdeg}(v) ::= |\{e \in E(G) \mid \text{tail}(e) = v\}|$$

An immediate consequence of this definition is

Lemma 10.1.2.

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

Proof. Both sums are equal to $|E(G)|$. ■

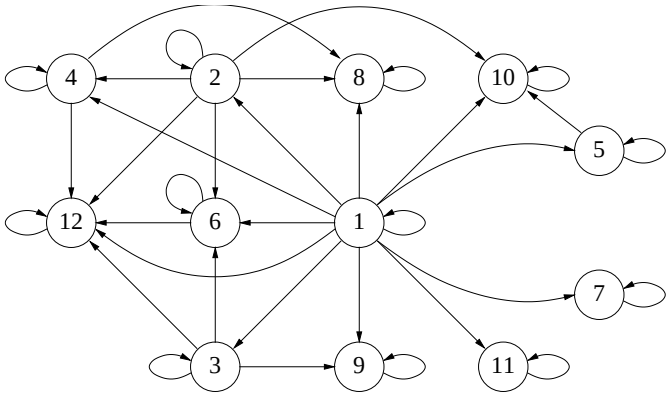


Figure 10.5 The Digraph for Divisibility on $\{1, 2, \ldots, 12\}$.

10.2 Walks and Paths

Picturing digraphs with points and arrows makes it natural to talk about following successive edges through the graph. For example, in the digraph of Figure 10.5, you might start at vertex 1, successively follow the edges from vertex 1 to vertex 2, from 2 to 4, from 4 to 12, and then from 12 to 12 twice (or as many times as you like). The sequence of edges followed in this way is called a *walk* through the graph. A *path* is a walk which never visits a vertex more than once. So following edges from 1 to 2 to 4 to 12 is a path, but it stops being a path if you go to 12 again.

The natural way to represent a walk is with the sequence of sucessive vertices it went through, in this case:

$$1 \;\; 2 \;\; 4 \;\; 12 \;\; 12 \;\; 12.$$

However, it is conventional to represent a walk by an alternating sequence of successive vertices and edges, so this walk would formally be

$$1 \;\; \langle 1 \to 2 \rangle \;\; 2 \;\; \langle 2 \to 4 \rangle \;\; 4 \;\; \langle 4 \to 12 \rangle \;\; 12 \;\; \langle 12 \to 12 \rangle \;\; 12 \;\; \langle 12 \to 12 \rangle \;\; 12. \tag{10.1}$$

The redundancy of this definition is enough to make any computer scientist cringe, but it does make it easy to talk about how many times vertices and edges occur on the walk. Here is a formal definition:

Definition 10.2.1. A *walk in a digraph* is an alternating sequence of vertices and edges that begins with a vertex, ends with a vertex, and such that for every edge $\langle u \to v \rangle$ in the walk, vertex u is the element just before the edge, and vertex v is the next element after the edge.

So a walk **v** is a sequence of the form

$$\mathbf{v} ::= v_0 \ \langle v_0 \to v_1 \rangle \ v_1 \ \langle v_1 \to v_2 \rangle \ v_2 \ \ldots \ \langle v_{k-1} \to v_k \rangle \ v_k$$

where $\langle v_i \to v_{i+1} \rangle \in E(G)$ for $i \in [0..k)$. The walk is said to *start* at v_0, to *end* at v_k, and the *length* $|\mathbf{v}|$ of the walk is defined to be k.

The walk is a *path* iff all the v_i's are different, that is, if $i \neq j$, then $v_i \neq v_j$.

A *closed walk* is a walk that begins and ends at the same vertex. A *cycle* is a positive length closed walk whose vertices are distinct except for the beginning and end vertices.

Note that a single vertex counts as a length zero path that begins and ends at itself. It also is a closed walk, but does not count as a cycle, since cycles by definition must have positive length. Length one cycles are possible when a node has an arrow leading back to itself. The graph in Figure 10.1 has none, but every vertex in the divisibility relation digraph of Figure 10.5 is in a length one cycle. Length one cycles are sometimes called *self-loops*.

Although a walk is officially an alternating sequence of vertices and edges, it is completely determined just by the sequence of successive vertices on it, or by the sequence of edges on it. We will describe walks in these ways whenever it's convenient. For example, for the graph in Figure 10.1,

- (a, b, d), or simply abd, is a (vertex-sequence description of a) length two path,
- $(\langle a \to b \rangle, \langle b \to d \rangle)$, or simply $\langle a \to b \rangle \langle b \to d \rangle$, is (an edge-sequence description of) the same length two path,
- $abcbd$ is a length four walk,
- $dcbcbd$ is a length five closed walk,
- $bdcb$ is a length three cycle,
- $\langle b \to c \rangle \langle c \to b \rangle$ is a length two cycle, and
- $\langle c \to b \rangle \langle b \leftarrow a \rangle \langle a \to d \rangle$ is *not* a walk. A walk is not allowed to follow edges in the wrong direction.

If you walk for a while, stop for a rest at some vertex, and then continue walking, you have broken a walk into two parts. For example, stopping to rest after following two edges in the walk (10.1) through the divisibility graph breaks the walk into the first part of the walk

$$1 \ \langle 1 \to 2 \rangle \ 2 \ \langle 2 \to 4 \rangle \ 4 \tag{10.2}$$

from 1 to 4, and the rest of the walk

$$4\ \langle 4\to 12\rangle\ 12\ \langle 12\to 12\rangle\ 12\ \langle 12\to 12\rangle\ 12. \tag{10.3}$$

from 4 to 12, and we'll say the whole walk (10.1) is the *merge* of the walks (10.2) and (10.3). In general, if a walk $\mathbf{f}$ ends with a vertex v and a walk $\mathbf{r}$ starts with the same vertex v we'll say that their *merge* $\mathbf{f}\widehat{\ }\mathbf{r}$ is the walk that starts with $\mathbf{f}$ and continues with $\mathbf{r}$.[1] Two walks can only be merged if the first walk ends at the same vertex v with which the second one walk starts. Sometimes it's useful to name the node v where the walks merge; we'll use the notation $\mathbf{f}\,\widehat{v}\,\mathbf{r}$ to describe the merge of a walk $\mathbf{f}$ that ends at v with a walk $\mathbf{r}$ that begins at v.

A consequence of this definition is that

Lemma 10.2.2.

$$|\mathbf{f}\widehat{\ }\mathbf{r}| = |\mathbf{f}| + |\mathbf{r}|.$$

In the next section we'll get mileage out of walking this way.

10.2.1 Finding a Path

If you were trying to walk somewhere quickly, you'd know you were in trouble if you came to the same place twice. This is actually a basic theorem of graph theory.

Theorem 10.2.3. *A shortest walk from one vertex to another is a path.*

Proof. If there is a walk from vertex u to another vertex $v \neq u$, then by the Well Ordering Principle, there must be a minimum length walk $\mathbf{w}$ from u to v. We claim $\mathbf{w}$ is a path.

To prove the claim, suppose to the contrary that $\mathbf{w}$ is not a path, meaning that some vertex x occurs twice on this walk. That is,

$$\mathbf{w} = \mathbf{e}\,\widehat{x}\,\mathbf{f}\,\widehat{x}\,\mathbf{g}$$

for some walks $\mathbf{e}, \mathbf{f}, \mathbf{g}$ where the length of $\mathbf{f}$ is positive. But then "deleting" $\mathbf{f}$ yields a strictly shorter walk

$$\mathbf{e}\,\widehat{x}\,\mathbf{g}$$

from u to v, contradicting the minimality of $\mathbf{w}$. ■

Definition 10.2.4. The *distance*, dist (u, v), in a graph from vertex u to vertex v is the length of a shortest path from u to v.

[1] It's tempting to say the *merge* is the concatenation of the two walks, but that wouldn't quite be right because if the walks were concatenated, the vertex v would appear twice in a row where the walks meet.

As would be expected, this definition of distance satisfies:

Lemma 10.2.5. *[The Triangle Inequality]*

$$dist(u, v) \leq dist(u, x) + dist(x, v)$$

for all vertices u, v, x with equality holding iff x is on a shortest path from u to v.

Of course, you might expect this property to be true, but distance has a technical definition and its properties can't be taken for granted. For example, unlike ordinary distance in space, the distance from u to v is typically different from the distance from v to u. So, let's prove the Triangle Inequality

Proof. To prove the inequality, suppose $\mathbf{f}$ is a shortest path from u to x and $\mathbf{r}$ is a shortest path from x to v. Then by Lemma 10.2.2, $\mathbf{f} \,\widehat{x}\, \mathbf{r}$ is a walk of length $\text{dist}(u, x) + \text{dist}(x, v)$ from u to v, so this sum is an upper bound on the length of the shortest path from u to v by Theorem 10.2.3.

Proof of the "iff" is in Problem 10.3. ■

Finally, the relationship between walks and paths extends to closed walks and cycles:

Lemma 10.2.6. *The shortest positive length closed walk through a vertex is a cycle through that vertex.*

The proof of Lemma 10.2.6 is essentially the same as for Theorem 10.2.3; see Problem 10.4.

10.3 Adjacency Matrices

If a graph G has n vertices $v_0, v_1, \ldots, v_{n-1}$, a useful way to represent it is with an $n \times n$ matrix of zeroes and ones called its *adjacency matrix* A_G. The ijth entry of the adjacency matrix, $(A_G)_{ij}$, is 1 if there is an edge from vertex v_i to vertex v_j and 0 otherwise. That is,

$$(A_G)_{ij} ::= \begin{cases} 1 & \text{if } \langle v_i \to v_j \rangle \in E(G), \\ 0 & \text{otherwise.} \end{cases}$$

For example, let H be the 4-node graph shown in Figure 10.1. Its adjacency matrix A_H is the 4×4 matrix:

$$A_H = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & 1 & 0 & 1 \\ b & 0 & 0 & 1 & 1 \\ c & 0 & 1 & 0 & 0 \\ d & 0 & 0 & 1 & 0 \end{array}$$

A payoff of this representation is that we can use matrix powers to count numbers of walks between vertices. For example, there are two length two walks between vertices a and c in the graph H:

$$a \ \langle a \to b \rangle \ b \ \langle b \to c \rangle \ c$$
$$a \ \langle a \to d \rangle \ d \ \langle d \to c \rangle \ c$$

and these are the only length two walks from a to c. Also, there is exactly one length two walk from b to c and exactly one length two walk from c to c and from d to b, and these are the only length two walks in H. It turns out we could have read these counts from the entries in the matrix $(A_H)^2$:

$$(A_H)^2 = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & 0 & 2 & 1 \\ b & 0 & 1 & 1 & 0 \\ c & 0 & 0 & 1 & 1 \\ d & 0 & 1 & 0 & 0 \end{array}$$

More generally, the matrix $(A_G)^k$ provides a count of the number of length k walks between vertices in any digraph G as we'll now explain.

Definition 10.3.1. The length-k *walk counting matrix* for an n-vertex graph G is the $n \times n$ matrix C such that

$$C_{uv} ::= \text{the number of length-}k\text{ walks from } u \text{ to } v. \tag{10.4}$$

Notice that the adjacency matrix A_G is the length-1 walk counting matrix for G, and that $(A_G)^0$, which by convention is the identity matrix, is the length-0 walk counting matrix.

Theorem 10.3.2. *If C is the length-k walk counting matrix for a graph G, and D is the length-m walk counting matrix, then CD is the length $k + m$ walk counting matrix for G.*

According to this theorem, the square $(A_G)^2$ of the adjacency matrix is the length two walk counting matrix for G. Applying the theorem again to $(A_G)^2 A_G$ shows that the length-3 walk counting matrix is $(A_G)^3$. More generally, it follows by induction that

Corollary 10.3.3. *The length-k counting matrix of a digraph G is $(A_G)^k$, for all $k \in \mathbb{N}$.*

In other words, you can determine the number of length k walks between any pair of vertices simply by computing the kth power of the adjacency matrix!

That may seem amazing, but the proof uncovers this simple relationship between matrix multiplication and numbers of walks.

Proof of Theorem 10.3.2. Any length $(k+m)$ walk between vertices u and v begins with a length k walk starting at u and ending at some vertex w followed by a length m walk starting at w and ending at v. So the number of length $(k+m)$ walks from u to v that go through w at the kth step equals the number C_{uw} of length k walks from u to w, times the number D_{wv} of length m walks from w to v. We can get the total number of length $(k+m)$ walks from u to v by summing, over all possible vertices w, the number of such walks that go through w at the kth step. In other words,

$$\#\text{length } (k+m) \text{ walks from } u \text{ to } v = \sum_{w \in V(G)} C_{uw} \cdot D_{wv} \tag{10.5}$$

But the right-hand side of (10.5) is precisely the definition of $(CD)_{uv}$. Thus, CD is indeed the length-$(k+m)$ walk counting matrix. ■

10.3.1 Shortest Paths

The relation between powers of the adjacency matrix and numbers of walks is cool—to us math nerds at least—but a much more important problem is finding shortest paths between pairs of nodes. For example, when you drive home for vacation, you generally want to take the shortest-time route.

One simple way to find the lengths of all the shortest paths in an n-vertex graph G is to compute the successive powers of A_G one by one up to the $n-1$st, watching for the first power at which each entry becomes positive. That's because Theorem 10.3.2 implies that the length of the shortest path, if any, between u and v, that is, the distance from u to v, will be the smallest value k for which $(A_G)^k_{uv}$ is nonzero, and if there is a shortest path, its length will be $\leq n-1$. Refinements of this idea lead to methods that find shortest paths in reasonably efficient ways. The methods apply as well to weighted graphs, where edges are labelled with weights or costs and the objective is to find least weight, cheapest paths. These refinements

are typically covered in introductory algorithm courses, and we won't go into them any further.

10.4 Walk Relations

A basic question about a digraph is whether there is a way to get from one particular vertex to another. So for any digraph G we are interested in a binary relation G^*, called the *walk relation* on $V(G)$, where

$$u\ G^*\ v ::= \text{there is a walk in } G \text{ from } u \text{ to } v. \tag{10.6}$$

Similarly, there is a *positive walk relation*

$$u\ G^+\ v ::= \text{there is a positive length walk in } G \text{ from } u \text{ to } v. \tag{10.7}$$

Definition 10.4.1. When there is a walk from vertex v to vertex w, we say that w is *reachable* from v, or equivalently, that v is *connected* to w.

10.4.1 Composition of Relations

There is a simple way to extend composition of functions to composition of relations, and this gives another way to talk about walks and paths in digraphs.

Definition 10.4.2. Let $R : B \to C$ and $S : A \to B$ be binary relations. Then the composition of R with S is the binary relation $(R \circ S) : A \to C$ defined by the rule

$$a\ (R \circ S)\ c ::= \exists b \in B.\ (a\ S\ b) \text{ AND } (b\ R\ c). \tag{10.8}$$

This agrees with the Definition 4.3.1 of composition in the special case when R and S are functions.[2]

Remembering that a digraph is a binary relation on its vertices, it makes sense to compose a digraph G with itself. Then if we let G^n denote the composition of G with itself n times, it's easy to check (see Problem 10.11) that G^n is the *length-n walk relation*:

$$a\ G^n\ b \quad \text{iff} \quad \text{there is a length } n \text{ walk in } G \text{ from } a \text{ to } b.$$

[2]The reversal of the order of R and S in (10.8) is not a typo. This is so that relational composition generalizes function composition. The value of function f composed with function g at an argument x is $f(g(x))$. So in the composition $f \circ g$, the function g is applied first.

This even works for $n = 0$, with the usual convention that G^0 is the *identity relation* $\mathrm{Id}_{V(G)}$ on the set of vertices.[3] Since there is a walk iff there is a path, and every path is of length at most $|V(G)| - 1$, we now have[4]

$$G^* = G^0 \cup G^1 \cup G^2 \cup \ldots \cup G^{|V(G)|-1} = (G \cup G^0)^{|V(G)|-1}. \tag{10.9}$$

The final equality points to the use of repeated squaring as a way to compute G^* with $\log n$ rather than $n - 1$ compositions of relations.

10.5 Directed Acyclic Graphs & Scheduling

Some of the prerequisites of MIT computer science subjects are shown in Figure 10.6. An edge going from subject s to subject t indicates that s is listed in the catalogue as a direct prerequisite of t. Of course, before you can take subject t, you have to take not only subject s, but also all the prerequisites of s, and any prerequisites of those prerequisites, and so on. We can state this precisely in terms of the positive walk relation: if D is the direct prerequisite relation on subjects, then subject u has to be completed before taking subject v iff $u\ D^+\ v$.

Of course it would take forever to graduate if this direct prerequisite graph had a positive length closed walk. We need to forbid such closed walks, which by Lemma 10.2.6 is the same as forbidding cycles. So, the direct prerequisite graph among subjects had better be *acyclic*:

Definition 10.5.1. A *directed acyclic graph (DAG)* is a directed graph with no cycles.

DAGs have particular importance in computer science. They capture key concepts used in analyzing task scheduling and concurrency control. When distributing a program across multiple processors, we're in trouble if one part of the program needs an output that another part hasn't generated yet! So let's examine DAGs and their connection to scheduling in more depth.

[3] The *identity relation* Id_A on a set A is the equality relation:

$$a\ \mathrm{Id}_A\ b \quad \text{iff} \quad a = b,$$

for $a, b \in A$.

[4] Equation (10.9) involves a harmless abuse of notation: we should have written

$$\mathrm{graph}(G^*) = \mathrm{graph}(G^0) \cup \mathrm{graph}(G^1) \ldots .$$

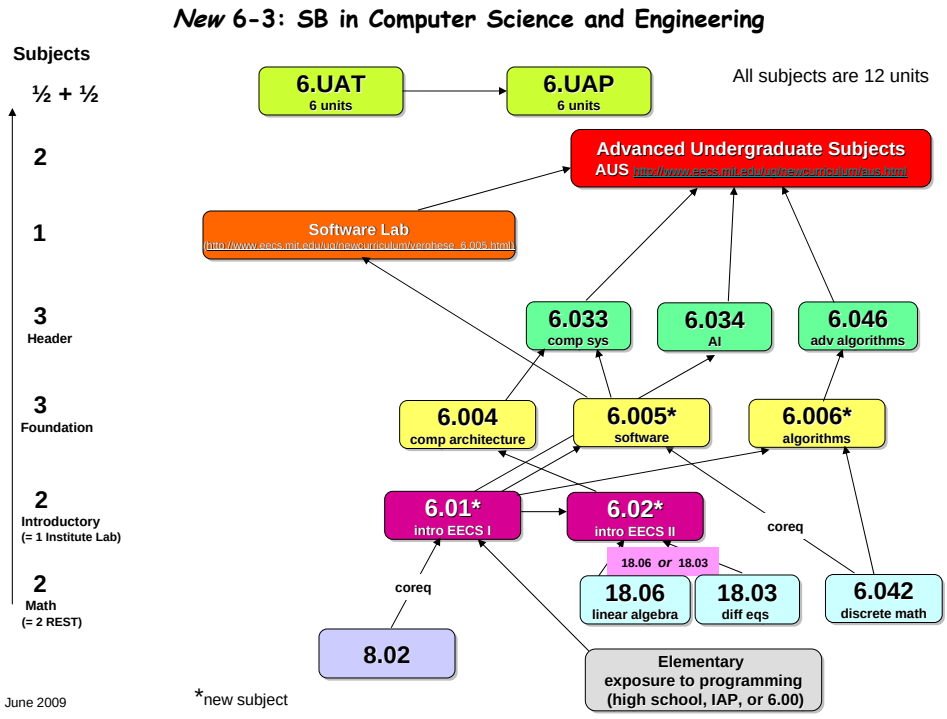


Figure 10.6 Subject prerequisites for MIT Computer Science (6-3) Majors.

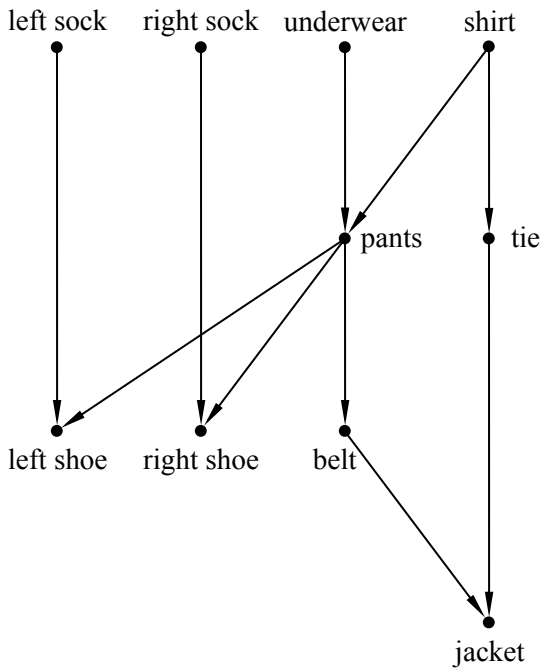


Figure 10.7 DAG describing which garments to put on before others.

10.5.1 Scheduling

In a scheduling problem, there is a set of tasks, along with a set of constraints specifying that starting certain tasks depends on other tasks being completed beforehand. We can map these sets to a digraph, with the tasks as the nodes and the direct prerequisite constraints as the edges.

For example, the DAG in Figure 10.7 describes how a man might get dressed for a formal occasion. As we describe above, vertices correspond to garments and edges specify which garments have to be put on before others.

When faced with a set of prerequisites like this one, the most basic task is finding an order in which to perform all the tasks, one at a time, while respecting the dependency constraints. Ordering tasks in this way is known as *topological sorting*.

Definition 10.5.2. A *topological sort* of a finite DAG is a list of all the vertices such that each vertex v appears earlier in the list than every other vertex reachable from v.

There are many ways to get dressed one item at a time while obeying the constraints of Figure 10.7. We have listed two such topological sorts in Figure 10.8. In

underwear	left sock
shirt	shirt
pants	tie
belt	underwear
tie	right sock
jacket	pants
left sock	right shoe
right sock	belt
left shoe	jacket
right shoe	left shoe
(a)	(b)

Figure 10.8 Two possible topological sorts of the prerequisites described in Figure 10.7

.

fact, we can prove that *every* finite DAG has a topological sort. You can think of this as a mathematical proof that you can indeed get dressed in the morning.

Topological sorts for finite DAGs are easy to construct by starting from *minimal* elements:

Definition 10.5.3. An vertex v of a DAG D is *minimum* iff every other vertex is reachable from v.

A vertex v is *minimal* iff v is not reachable from any other vertex.

It can seem peculiar to use the words "minimum" and "minimal" to talk about vertices that start paths. These words come from the perspective that a vertex is "smaller" than any other vertex it connects to. We'll explore this way of thinking about DAGs in the next section, but for now we'll use these terms because they are conventional.

One peculiarity of this terminology is that a DAG may have no minimum element but lots of minimal elements. In particular, the clothing example has four minimal elements: leftsock, rightsock, underwear, and shirt.

To build an order for getting dressed, we pick one of these minimal elements—say, shirt. Now there is a new set of minimal elements; the three elements we didn't chose as step 1 are still minimal, and once we have removed shirt, tie becomes minimal as well. We pick another minimal element, continuing in this way 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 sorts above were constructed.

So our construction shows:

Theorem 10.5.4. *Every finite DAG has a topological sort.*

There are many other ways of constructing topological sorts. For example, instead of starting from the minimal elements at the beginning of paths, we could build a topological sort starting from *maximal* elements at the end of paths. In fact, we could build a topological sort by picking vertices arbitrarily from a finite DAG and simply inserting them into the list wherever they will fit.[5]

10.5.2 Parallel Task Scheduling

For task dependencies, topological sorting provides a way to execute tasks one after another while respecting those 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 DAG 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 set of tasks, how long does it take to do them all in an optimal parallel schedule? We can use walk relations on acyclic graphs to analyze this problem.

In the first unit of time, we should do all minimal items, so we would put on our left sock, our right sock, our underwear, and our shirt.[6] In the second unit of time, we should put on our pants and our tie. Note that we cannot put on our left or right shoe yet, since we have not yet put on our pants. In the third unit of time, we should put on our left shoe, our right shoe, and our belt. Finally, in the last unit of time, we can put on our jacket. This schedule is illustrated in Figure 10.9.

The total time to do these tasks is 4 units. We cannot do better than 4 units of time because there is a sequence of 4 tasks that must each be done before the next. We have to put on a shirt before pants, pants before a belt, and a belt before a jacket. Such a sequence of items is known as a *chain*.

Definition 10.5.5. Two vertices in a DAG are *comparable* when one of them is reachable from the other. A *chain* in a DAG is a set of vertices such that any two of them are comparable. A vertex in a chain that is reachable from all other vertices in the chain is called a *maximum element* of the chain. A finite chain is said to *end at* its maximum element.

[5] Topolgical sorts can be generalized and shown to exist even for infinite DAGs, but you'll be relieved to know that we have no need to go into this.

[6] Yes, we know that you can't actually put on both socks at once, but imagine you are being dressed by a bunch of robot processors and you are in a big hurry. Still not working for you? Ok, forget about the clothes and imagine they are programs with the precedence constraints shown in Figure 10.7.

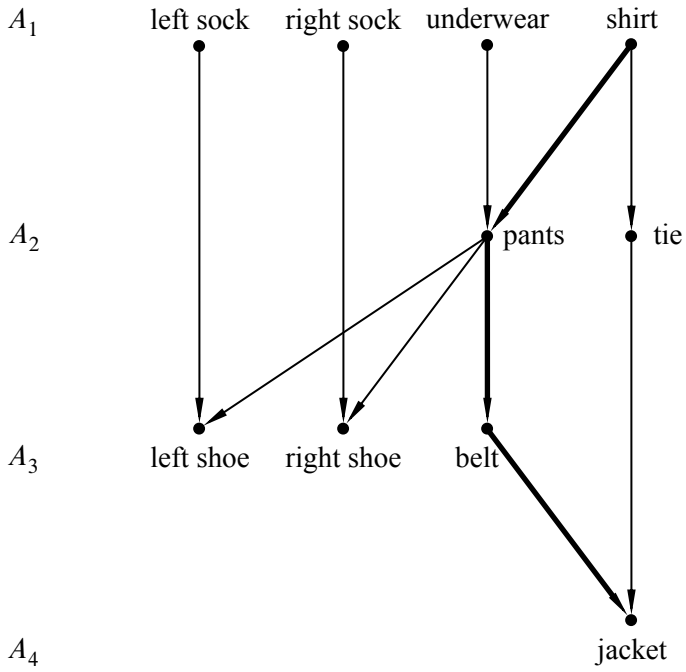


Figure 10.9 A parallel schedule for the tasks-getting-dressed digraph in Figure 10.7. The tasks in A_i can be performed in step i for $1 \leq i \leq 4$. A chain of 4 tasks (the critical path in this example) is shown with bold edges.

The time it takes to schedule tasks, even with an unlimited number of processors, is at least as large as the number of vertices in any chain. That's because if we used less time than the size of some chain, then two items from the chain would have to be done at the same step, contradicting the precedence constraints. For this reason, a *largest* chain is also known as a *critical path*. For example, Figure 10.9 shows the critical path for the getting-dressed digraph.

In this example, we were able to schedule all the tasks with t steps, where t is the size of the largest chain. A nice feature of DAGs is that this is always possible! In other words, for any DAG, there is a legal parallel schedule that runs in t total steps.

In general, a *schedule* for performing tasks specifies which tasks to do at successive steps. Every task a has to be scheduled at some step, and all the tasks that have to be completed before task a must be scheduled for an earlier step.

Let's be precise about the definition of schedule.

Definition 10.5.6. A *partition* of a set A is a set of nonempty subsets of A called the *blocks*[7] of the partition, such that every element of A is in exactly one block.

For example, one possible partition of the set $\{a, b, c, d, e\}$ into three blocks is

$$\{a, c\} \qquad \{b, e\} \qquad \{d\}.$$

Definition 10.5.7. A *parallel schedule* for a DAG D is a partition of $V(D)$ into blocks $A_0, A_1, \ldots$, such that when $j < k$, no vertex in A_j is reachable from any vertex in A_k. The block A_k is called the set of elements *scheduled at step* k, and the *time* of the schedule is the number of blocks. The maximum number of elements scheduled at any step is called the *number of processors* required by the schedule.

A *largest* chain ending at an element a is called a *critical path* to a, and the number of elements less than a in the chain is called the *depth* of a. So in any possible parallel schedule, there must be at least depth (a) steps before task a can be started. In particular, the minimal elements are precisely the elements with depth 0.

There is a very simple schedule that completes every task in its minimum number of steps: just use a "greedy" strategy of performing tasks as soon as possible. Schedule all the elements of depth k at step k. That's how we found the above schedule for getting dressed.

Theorem 10.5.8. *A minimum time schedule for a finite DAG D consists of the sets $A_0, A_1, \ldots$, where*

$$A_k ::= \{a \in V(D) \mid \text{depth}\,(a) = k\}.$$

[7] We think it would be nicer to call them the *parts* of the partition, but "blocks" is the standard terminology.

We'll leave to Problem 10.25 the proof that the sets A_k are a parallel schedule according to Definition 10.5.7. We can summarize the story above in this way: with an unlimited number of processors, the parallel time to complete all tasks is simply the size of a critical path:

Corollary 10.5.9. *Parallel time = size of critical path.*

Things get more complex when the number of processors is bounded; see Problem 10.26 for an example.

10.5.3 Dilworth's Lemma

Definition 10.5.10. An *antichain* in a DAG is a set of vertices such that *no* two elements in the set are comparable—no walk exists between any two different vertices in the set.

Our conclusions about scheduling also tell us something about antichains.

Corollary 10.5.11. *In a DAG D if the size of the largest chain is t, then $V(D)$ can be partitioned into t antichains.*

Proof. Let the antichains be the sets $A_k ::= \{a \in V(D) \mid \text{depth}(a) = k\}$. It is an easy exercise to verify that each A_k is an antichain (Problem 10.25). ■

Corollary 10.5.11 implies[8] a famous result about acyclic digraphs:

Lemma 10.5.12 (Dilworth)**.** *For all $t > 0$, every DAG with n vertices must have either a chain of size greater than t or an antichain of size at least n/t.*

Proof. Assume that there is no chain of size greater than t. Let ℓ be the size of the largest antichain. If we make a parallel schedule according to the proof of Corollary 10.5.11, we create a number of antichains equal to the size of the largest chain, which is less than or equal t. Each element belongs to exactly one antichain, none of which are larger than ℓ. So the total number of elements is at most ℓ times t—that is, $\ell t \geq n$. Simple division implies that $\ell \geq n/t$. ■

Corollary 10.5.13. *Every DAG with n vertices 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 10.5.12. ■

Example 10.5.14. When the man in our example is getting dressed, $n = 10$.

Try $t = 3$. There is a chain of size 4.

Try $t = 4$. There is no chain of size 5, but there is an antichain of size $4 \geq 10/4$.

[8] Lemma 10.5.12 also follows from a more general result known as Dilworth's Theorem, which we will not discuss.

10.6 Partial Orders

After mapping the "direct prerequisite" relation onto a digraph, we were then able to use the tools for understanding computer scientists' graphs to make deductions about something as mundane as getting dressed. This may or may not have impressed you, but we can do better. In the introduction to this chapter, we mentioned a useful fact that bears repeating: any digraph is formally the same as a binary relation whose domain and codomain are its vertices. This means that *any* binary relation whose domain is the same as its codomain can be translated into a digraph! Talking about the edges of a binary relation or the image of a set under a digraph may seem odd at first, but doing so will allow us to draw important connections between different types of relations. For instance, we can apply Dilworth's lemma to the "direct prerequisite" relation for getting dressed, because the graph of that relation was a DAG.

But how can we tell if a binary relation is a DAG? And once we know that a relation is a DAG, what exactly can we conclude? In this section, we will abstract some of the properties that a binary relation might have, and use those properties to define classes of relations. In particular, we'll explain this section's title, *partial orders*.

10.6.1 The Properties of the Walk Relation in DAGs

To begin, let's talk about some features common to all digraphs. Since merging a walk from u to v with a walk from v to w gives a walk from u to w, both the walk and positive walk relations have a relational property called *transitivity*:

Definition 10.6.1. A binary relation R on a set A is *transitive* iff

$$(a\ R\ b \text{ AND } b\ R\ c) \text{ IMPLIES } a\ R\ c$$

for every $a, b, c \in A$.

So we have

Lemma 10.6.2. *For any digraph G the walk relations G^+ and G^* are transitive.*

Since there is a length zero walk from any vertex to itself, the walk relation has another relational property called *reflexivity*:

Definition 10.6.3. A binary relation R on a set A is *reflexive* iff $a\ R\ a$ for all $a \in A$.

Now we have

Lemma 10.6.4. *For any digraph G, the walk relation G^* is reflexive.*

We know that a digraph is a DAG iff it has no positive length closed walks. Since any vertex on a closed walk can serve as the beginning and end of the walk, saying a graph is a DAG is the same as saying that there is no positive length path from any vertex back to itself. This means that the positive walk relation D^+ of a DAG has a relational property called *irreflexivity*.

Definition 10.6.5. A binary relation R on a set A is *irreflexive* iff

$$\text{NOT}(a\ R\ a)$$

for all $a \in A$.

So we have

Lemma 10.6.6. *R is a DAG iff R^+ is irreflexive.*

10.6.2 Strict Partial Orders

Here is where we begin to define interesting classes of relations:

Definition 10.6.7. A relation that is transitive and irreflexive is called a *strict partial order*.

A simple connection between strict partial orders and DAGs now follows from Lemma 10.6.6:

Theorem 10.6.8. *A relation R is a strict partial order iff R is the positive walk relation of a DAG.*

Strict partial orders come up in many situations which on the face of it have nothing to do with digraphs. For example, the less-than order $<$ on numbers is a strict partial order:

- if $x < y$ and $y < z$ then $x < z$, so less-than is transitive, and
- $\text{NOT}(x < x)$, so less-than is irreflexive.

The proper containment relation $\subset$ is also a partial order:

- if $A \subset B$ and $B \subset C$ then $A \subset C$, so containment is transitive, and
- $\text{NOT}(A \subset A)$, so proper containment is irreflexive.

If there are two vertices that are reachable from each other, then there is a positive length closed walk that starts at one vertex, goes to the other, and then comes back. So DAGs are digraphs in which no two vertices are mutually reachable. This corresponds to a relational property called *asymmetry*.

Definition 10.6.9. A binary relation R on a set A is *asymmetric* iff

$$a\ R\ b \text{ IMPLIES NOT}(b\ R\ a)$$

for all $a, b \in A$.

So we can also characterize DAGs in terms of asymmetry:

Corollary 10.6.10. *A digraph D is a DAG iff D^+ is asymmetric.*

Corollary 10.6.10 and Theorem 10.6.8 combine to give

Corollary 10.6.11. *A binary relation R on a set A is a strict partial order iff it is transitive and asymmetric.*[9]

A strict partial order may be the positive walk relation of different DAGs. This raises the question of finding a DAG with the *smallest* number of edges that determines a given strict partial order. For *finite* strict partial orders, the smallest such DAG turns out to be unique and easy to find (see Problem 10.31).

10.6.3 Weak Partial Orders

The less-than-or-equal relation $\leq$ is at least as familiar as the less-than strict partial order, and the ordinary containment relation $\subseteq$ is even more common than the proper containment relation. These are examples of *weak partial orders*, which are just strict partial orders with the additional condition that every element is related to itself. To state this precisely, we have to relax the asymmetry property so it does not apply when a vertex is compared to itself; this relaxed property is called *antisymmetry*:

Definition 10.6.12. A binary relation R on a set A, is *antisymmetric* iff, for all $a \neq b \in A$,

$$a\ R\ b \text{ IMPLIES NOT}(b\ R\ a)$$

Now we can give an axiomatic definition of weak partial orders that parallels the definition of strict partial orders.

[9] Some texts use this corollary to define strict partial orders.

Definition 10.6.13. A binary relation on a set is a *weak partial order* iff it is transitive, reflexive, and antisymmetric.

The following lemma gives another characterization of weak partial orders that follows directly from this definition.

Lemma 10.6.14. *A relation R on a set A is a weak partial order iff there is a strict partial order S on A such that*

$$a \; R \; b \quad \textit{iff} \quad (a \; S \; b \;\; \text{OR} \;\; a = b),$$

for all $a, b \in A$.

Since a length zero walk goes from a vertex to itself, this lemma combined with Theorem 10.6.8 yields:

Corollary 10.6.15. *A relation is a weak partial order iff it is the walk relation of a DAG.*

For weak partial orders in general, we often write an ordering-style symbol like $\preceq$ or $\sqsubseteq$ instead of a letter symbol like R.[10] Likewise, we generally use $\prec$ or $\sqsubset$ to indicate a strict partial order.

Two more examples of partial orders are worth mentioning:

Example 10.6.16. Let A be some family of sets and define $a \; R \; b$ iff $a \supset b$. Then R is a strict partial order.

Example 10.6.17. The divisibility relation is a weak partial order on the nonnegative integers.

For practice with the definitions, you can check that two more examples are vacuously partial orders on a set D: the identity relation Id_D is a weak partial order, and the *empty relation*—the relation with no arrows—is a strict partial order.

Note that some authors define "partial orders" to be what we call weak partial orders. However, we'll use the phrase "partial order" to mean a relation that may be either a weak or strict partial order.

10.7 Representing Partial Orders by Set Containment

Axioms can be a great way to abstract and reason about important properties of objects, but it helps to have a clear picture of the things that satisfy the axioms.

[10] General relations are usually denoted by a letter like R instead of a cryptic squiggly symbol, so $\preceq$ is kind of like the musical performer/composer Prince, who redefined the spelling of his name to be his own squiggly symbol. A few years ago he gave up and went back to the spelling "Prince."

DAGs provide one way to picture partial orders, but it also can help to picture them in terms of other familiar mathematical objects. In this section, we'll show that every partial order can be pictured as a collection of sets related by containment. That is, every partial order has the "same shape" as such a collection. The technical word for "same shape" is "isomorphic."

Definition 10.7.1. A binary relation R on a set A is *isomorphic* to a relation S on a set B iff there is a relation-preserving bijection from A to B; that is, there is a bijection $f : A \to B$ such that for all $a, a' \in A$,

$$a \; R \; a' \quad \text{iff} \quad f(a) \; S \; f(a').$$

To picture a partial order $\preceq$ on a set A as a collection of sets, we simply represent each element A by the set of elements that are $\preceq$ to that element, that is,

$$a \longleftrightarrow \{b \in A \mid b \preceq a\}.$$

For example, if $\preceq$ is the divisibility relation on the set of integers $\{1, 3, 4, 6, 8, 12\}$, then we represent each of these integers by the set of integers in A that divide it. So

$$\begin{aligned} 1 &\longleftrightarrow \{1\} \\ 3 &\longleftrightarrow \{1, 3\} \\ 4 &\longleftrightarrow \{1, 4\} \\ 6 &\longleftrightarrow \{1, 3, 6\} \\ 8 &\longleftrightarrow \{1, 4, 8\} \\ 12 &\longleftrightarrow \{1, 3, 4, 6, 12\} \end{aligned}$$

So, the fact that $3 \mid 12$ corresponds to the fact that $\{1, 3\} \subseteq \{1, 3, 4, 6, 12\}$.

In this way we have completely captured the weak partial order $\preceq$ by the subset relation on the corresponding sets. Formally, we have

Lemma 10.7.2. *Let $\preceq$ be a weak partial order on a set A. Then $\preceq$ is isomorphic to the subset relation $\subseteq$ on the collection of inverse images under the $\preceq$ relation of elements $a \in A$.*

We leave the proof to Problem 10.37. Essentially the same construction shows that strict partial orders can be represented by sets under the proper subset relation, $\subset$ (Problem 10.38). To summarize:

Theorem 10.7.3. *Every weak partial order $\preceq$ is isomorphic to the subset relation $\subseteq$ on a collection of sets.*

Every strict partial order $\prec$ is isomorphic to the proper subset relation $\subset$ on a collection of sets.

10.8 Linear Orders

The familiar order relations on numbers have an important additional property: given two different numbers, one will be bigger than the other. Partial orders with this property are said to be *linear orders*. You can think of a linear order as one where all the elements are lined up so that everyone knows exactly who is ahead and who is behind them in the line.[11]

Definition 10.8.1. 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 $[a \ R \ b \ \text{OR} \ b \ R \ a]$. A partial order for which every two different elements are comparable is called a *linear order*.

So $<$ and $\leq$ are linear orders on $\mathbb{R}$. On the other hand, the subset relation is *not* linear, since, for example, any two different finite sets of the same size will be incomparable under $\subseteq$. The prerequisite relation on Course 6 required subjects is also not linear because, for example, neither 8.01 nor 6.042 is a prerequisite of the other.

10.9 Product Orders

Taking the product of two relations is a useful way to construct new relations from old ones.

Definition 10.9.1. The product $R_1 \times R_2$ of relations R_1 and R_2 is defined to be the relation with

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

It follows directly from the definitions that products preserve the properties of transitivity, reflexivity, irreflexivity, and antisymmetry (see Problem 10.52). If R_1 and R_2 both have one of these properties, then so does $R_1 \times R_2$. This implies that if R_1 and R_2 are both partial orders, then so is $R_1 \times R_2$.

[11] Linear orders are often called "total" orders, but this terminology conflicts with the definition of "total relation," and it regularly confuses students.

Being a linear order is a much stronger condition than being a partial order that is a total relation. For example, any weak partial order is a total relation but generally won't be linear.

Example 10.9.2. 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 nonnegative integer ≤ 2400 that we interpret as an age in months, and h is a nonnegative integer ≤ 120 describing height in inches. We define Y by the rule

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

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

Since both ages and heights are ordered numerically, the age-height relation Y is a partial order. Now suppose we have a class of 101 students. Then we can apply Dilworth's lemma 10.5.12 to conclude that there is a chain of 11 students—that is, 11 students who get taller as they get older–or an antichain of 11 students—that is, 11 students who get taller as they get younger, which makes for an amusing in-class demo.

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

10.10 Equivalence Relations

Definition 10.10.1. A relation R on a set A is *symmetric* when

$$\forall x, y \in A.\ x\ R\ y \ \text{IMPLIES}\ y\ R\ x.$$

A relation is an *equivalence relation* iff it is reflexive, symmetric, and transitive.

Congruence modulo n is an important example of an equivalence relation:

- It is reflexive because $x \equiv x \pmod{n}$.
- It is symmetric because $x \equiv y \pmod{n}$ implies $y \equiv x \pmod{n}$.
- It is transitive because $x \equiv y \pmod{n}$ and $y \equiv z \pmod{n}$ imply that $x \equiv z \pmod{n}$.

There is an even more well-known example of an equivalence relation: equality itself.

If we think of a total function f as attaching a label to each element in its domain, then "having the same label" defines an equivalence relation. More precisely, if f is a total function, we can define a basic equivalence relation $\equiv_f$ on the domain of f as follows:

Definition 10.10.2. If f is a total function, the relation $\equiv_f$ on domain(f) is defined by the rule:

$$a \equiv_f b \text{ IFF } f(a) = f(b).$$

The relation $\equiv_f$ inherits the properties of reflexivity, symmetry and transitivity from the corresponding properties of the equality relation. So $\equiv_f$ is indeed an equivalence relation. This observation gives another way to see that congruence modulo n is an equivalence relation: the Remainder Lemma 9.6.1 implies that congruence modulo n is the same as $\equiv_r$ where $r(a)$ is the remainder of a divided by n.

In fact, a relation is an equivalence relation iff it equals $\equiv_f$ for some total function f (see Problem 10.58). So every equivalence relation can actually be understood as being the relation of "having the same label" according to some labelling scheme.

10.10.1 Equivalence Classes

Equivalence relations are closely related to partitions because the images of elements under an equivalence relation are the blocks of a partition.

Definition 10.10.3. Given an equivalence relation $R : A \to A$, the *equivalence class* $[a]_R$ of an element $a \in A$ is the set of all elements of A related to a by R. Namely,

$$[a]_R ::= \{x \in A \mid a \; R \; x\}.$$

In other words, $[a]_R$ is the image $R(a)$.

For example, suppose that $A = \mathbb{Z}$ and $a \; R \; b$ means that $a \equiv b \pmod 5$. Then

$$[7]_R = \{\ldots, -3, 2, 7, 12, 17, 22, \ldots\}.$$

Notice that each of the elements in $[7]_R$ defines have this same equivalence class; that is, $[7]_R = [-3]_R = [2]_R = [12]_R = \cdots$.

There is an exact correspondence between equivalence relations on A and partitions of A. Namely, given any partition of a set, being in the same block is obviously an equivalence relation. On the other hand we have:

Theorem 10.10.4. *The equivalence classes of an equivalence relation on a set A are the blocks of a partition of A.*

We'll leave the proof of Theorem 10.10.4 as a basic exercise in axiomatic reasoning (see Problem 10.57), but let's look at an example. The congruent-mod-5

relation partitions the integers into five equivalence classes:

$$\{\ldots, -5, 0, 5, 10, 15, 20, \ldots\}$$
$$\{\ldots, -4, 1, 6, 11, 16, 21, \ldots\}$$
$$\{\ldots, -3, 2, 7, 12, 17, 22, \ldots\}$$
$$\{\ldots, -2, 3, 8, 13, 18, 23, \ldots\}$$
$$\{\ldots, -1, 4, 9, 14, 19, 24, \ldots\}$$

In these terms, $x \equiv y \pmod 5$ is equivalent to the assertion that x and y are both in the same block of this partition. For example, $6 \equiv 16 \pmod 5$, because they're both in the second block, but $2 \not\equiv 9 \pmod 5$ because 2 is in the third block while 9 is in the last block.

In social terms, if "likes" were an equivalence relation, then everyone would be partitioned into cliques of friends who all like each other and no one else.

10.11 Summary of Relational Properties

A relation $R : A \to A$ is the same as a digraph with vertices A.

Reflexivity R is *reflexive* when

$$\forall x \in A.\ x\ R\ x.$$

Every vertex in R has a self-loop.

Irreflexivity R is *irreflexive* when

$$\text{NOT}[\exists x \in A.\ x\ R\ x].$$

There are no self-loops in R.

Symmetry R is *symmetric* when

$$\forall x, y \in A.\ x\ R\ y \text{ IMPLIES } y\ R\ x.$$

If there is an edge from x to y in R, then there is an edge back from y to x as well.

Asymmetry R is *asymmetric* when

$$\forall x, y \in A.\ x\ R\ y \text{ IMPLIES NOT}(y\ R\ x).$$

There is at most one directed edge between any two vertices in R, and there are no self-loops.

Antisymmetry R is *antisymmetric* when

$$\forall x \neq y \in A.\ x\ R\ y \text{ IMPLIES NOT}(y\ R\ x).$$

Equivalently,

$$\forall x, y \in A.\ (x\ R\ y \text{ AND } y\ R\ x) \text{ IMPLIES } x = y.$$

There is at most one directed edge between any two distinct vertices, but there may be self-loops.

Transitivity R is *transitive* when

$$\forall x, y, z \in A.\ (x\ R\ y \text{ AND } y\ R\ z) \text{ IMPLIES } x\ R\ z.$$

If there is a positive length path from u to v, then there is an edge from u to v.

Linear R is *linear* when

$$\forall x \neq y \in A.\ (x\ R\ y \text{ OR } y\ R\ x)$$

Given any two vertices in R, there is an edge in one direction or the other between them.

Strict Partial Order R is a *strict partial order* iff R is transitive and irreflexive iff R is transitive and asymmetric iff it is the positive length walk relation of a DAG.

Weak Partial Order R is a *weak partial order* iff R is transitive and anti-symmetric and reflexive iff R is the walk relation of a DAG.

Equivalence Relation R is an *equivalence relation* iff R is reflexive, symmetric and transitive iff R equals the *in-the-same-block*-relation for some partition of domain(R).

10.12 References

[9], [15], [24], [27], [29]

Problems for Section 10.1

Practice Problems

Problem 10.1.
Let S be a nonempty set of size $n \in \mathbb{Z}^+$, and let $f : S \to S$ be total function. Let D_f be the digraph with vertices S whose edges are $\{\langle s \to f(s)\rangle \mid s \in S\}$.

(a) What are the possible values of the out-degrees of vertices of D_f?

(b) What are the possible values of the in-degrees of the vertices?

(c) Suppose f is a surjection. Now what are the possible values of the in-degrees of the vertices?

Exam Problems

Problem 10.2.
The proof of the Handshaking Lemma 10.1.2 invoked the "obvious" fact that in any finite digraph, the sum of the in-degrees of the vertices equals the number of arrows in the graph. That is,

Claim. *For any finite digraph G*

$$\sum_{v \in V(G)} \text{indeg}(v) = |\,\text{graph}(G)|, \tag{10.10}$$

But this Claim might not be obvious to everyone. So prove it by induction on the number $|\,\text{graph}(G)|$ of arrows.

Problems for Section 10.2

Practice Problems

Problem 10.3.
Lemma 10.2.5 states that $\text{dist}(u, v) \leq \text{dist}(u, x) + \text{dist}(x, v)$. It also states that

equality holds iff x is on a shortest path from u to v.

(a) Prove the "iff" statement from left to right.

(b) Prove the "iff" from right to left.

Class Problems

Problem 10.4. **(a)** Give an example of a digraph that has a closed walk including two vertices but has no cycle including those vertices.

(b) Prove Lemma 10.2.6:

Lemma. *The shortest positive length closed walk through a vertex is a cycle.*

Problem 10.5.

A 3-bit string is a string made up of 3 characters, each a 0 or a 1. Suppose you'd like to write out, in one string, all eight of the 3-bit strings in any convenient order. For example, if you wrote out the 3-bit strings in the usual order starting with 000 001 010..., you could concatenate them together to get a length $3 \cdot 8 = 24$ string that started 000001010....

But you can get a shorter string containing all eight 3-bit strings by starting with 00010.... Now 000 is present as bits 1 through 3, and 001 is present as bits 2 through 4, and 010 is present as bits 3 through 5, ...

(a) Say a string is *3-good* if it contains every 3-bit string as 3 consecutive bits somewhere in it. Find a 3-good string of length 10, and explain why this is the minimum length for any string that is 3-good.

(b) Explain how any walk that includes every edge in the graph shown in Figure 10.10 determines a string that is 3-good. Find the walk in this graph that determines your 3-good string from part (a).

(c) Explain why a walk in the graph of Figure 10.10 that includes every every edge *exactly once* provides a minimum-length 3-good string.[12]

(d) Generalize the 2-bit graph to a k-bit digraph B_k for $k \geq 2$, where $V(B_k) ::= \{0,1\}^k$, and any walk through B_k that contains every edge exactly once determines a minimum length $(k+1)$-good bit-string.[13]

[12] The 3-good strings explained here generalize to n-good strings for $n \geq 3$. They were studied by the great Dutch mathematician/logician Nicolaas de Bruijn, and are known as *de Bruijn sequences*. de Bruijn died in February, 2012 at the age of 94.

[13] Problem 10.7 explains why such "Eulerian" paths exist.

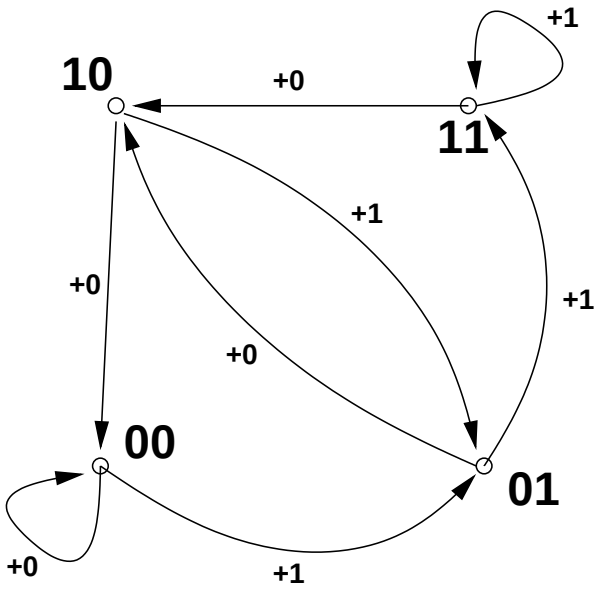


Figure 10.10 The 2-bit graph.

What is this minimum length?

Define the transitions of B_k. Verify that the in-degree of each vertex is the same as its out-degree and that there is a positive length path from any vertex to any other vertex (including itself) of length at most k.

Homework Problems

Problem 10.6. (a) Give an example of a digraph in which a vertex v is on a positive even-length closed walk, but *no* vertex is on an even-length cycle.

(b) Give an example of a digraph in which a vertex v is on an odd-length closed walk but not on an odd-length cycle.

(c) Prove that every odd-length closed walk contains a vertex that is on an odd-length cycle.

Problem 10.7.
An *Euler tour*[14] of a graph is a closed walk that includes every edge exactly once.

[14] In some other texts, this is called an *Euler circuit*.

Such walks are named after the famous 17th century mathematician Leonhard Euler. (Same Euler as for the constant $e \approx 2.718$ and the totient function ϕ—he did a lot of stuff.)

So how do you tell in general whether a graph has an Euler tour? At first glance this may seem like a daunting problem (the similar sounding problem of finding a cycle that touches every vertex exactly once is one of those million dollar NP-complete problems known as the *Hamiltonian Cycle Problem*)—but it turns out to be easy.

(a) Show that if a graph has an Euler tour, then the in-degree of each vertex equals its out-degree.

A digraph is *weakly connected* if there is a "path" between any two vertices that may follow edges backwards or forwards.[15] In the remaining parts, we'll work out the converse. Suppose a graph is weakly connected, and the in-degree of every vertex equals its out-degree. We will show that the graph has an Euler tour.

A *trail* is a walk in which each edge occurs *at most* once.

(b) Suppose that a trail in a weakly connected graph does not include every edge. Explain why there must be an edge not on the trail that starts or ends at a vertex on the trail.

In the remaining parts, assume the graph is weakly connected, and the in-degree of every vertex equals its out-degree. Let **w** be the *longest* trail in the graph.

(c) Show that if **w** is closed, then it must be an Euler tour.

Hint: part (b)

(d) Explain why all the edges starting at the end of **w** must be on **w**.

(e) Show that if **w** was not closed, then the in-degree of the end would be bigger than its out-degree.

Hint: part (d)

(f) Conclude that if the in-degree of every vertex equals its out-degree in a finite, weakly connected digraph, then the digraph has an Euler tour.

[15] More precisely, a graph G is weakly connected iff there is a path from any vertex to any other vertex in the graph H with

$$V(H) = V(G), \text{ and}$$
$$E(H) = E(G) \cup \{\langle v \to u\rangle \mid \langle u \to v\rangle \in E(G)\}.$$

In other words $H = G \cup G^{-1}$.

Problems for Section 10.3

Homework Problems

Problem 10.8.
The *weight of a walk* in a weighted graph is the sum of the weights of the successive edges in the walk. The *minimum weight matrix* for length k walks in an n-vertex graph G is the $n \times n$ matrix W such that for $u, v \in V(G)$,

$$W_{uv} ::= \begin{cases} w & \text{if } w \text{ is the minimum weight among length } k \text{ walks from } u \text{ to } v, \\ \infty & \text{if there is no length } k \text{ walk from } u \text{ to } v. \end{cases}$$

The min+ product of two $n \times n$ matrices W and M with entries in $\mathbb{R} \cup \{\infty\}$ is the $n \times n$ matrix $W \underset{\min+}{\cdot} M$ whose ij entry is

$$(W \underset{\min+}{\cdot} V)_{ij} ::= \min\{W_{ik} + V_{kj} \mid 1 \le k \le n\}.$$

Prove the following theorem.

Theorem. *If W is the minimum weight matrix for length k walks in a weighted graph G, and V is the minimum weight matrix for length m walks, then $W \underset{min+}{\cdot} V$ is the minimum weight matrix for length $k + m$ walks.*

Problems for Section 10.4

Practice Problems

Problem 10.9.
Let

$$A ::= \{1, 2, 3\}$$
$$B ::= \{4, 5, 6\}$$
$$R ::= \{(1, 4), (1, 5), (2, 5), (3, 6)\}$$
$$S ::= \{(4, 5), (4, 6), (5, 4)\}.$$

Note that R is a relation from A to B and S is a relation from B to B.

List the pairs in each of the relations below.

(a) $S \circ R$.

(b) $S \circ S$.

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

Problem 10.10.
In a round-robin tournament, every two distinct players play against each other just once. For a round-robin tournament with no tied games, a record of who beat whom can be described with a *tournament digraph*, where the vertices correspond to players and there is an edge $\langle x \to y \rangle$ iff x beat y in their game.

A *ranking* is a path that includes all the players. So in a ranking, each player won the game against the next lowest ranked player, but may very well have lost their games against much lower ranked players—whoever does the ranking may have a lot of room to play favorites.

(a) Give an example of a tournament digraph with more than one ranking.

(b) Prove that if a tournament digraph is a DAG, then it has at most one ranking.

(c) Prove that every finite tournament digraph has a ranking.

Homework Problems

Problem 10.11.
Let R be a binary relation on a set A. Regarding R as a digraph, let $W^{(n)}$ denote the length-n walk relation in the digraph R, that is,

$$a\ W^{(n)}\ b ::= \text{there is a length } n \text{ walk from } a \text{ to } b \text{ in } R.$$

(a) Prove that

$$W^{(n)} \circ W^{(m)} = W^{(m+n)} \tag{10.11}$$

for all $m, n \in \mathbb{N}$, where $\circ$ denotes relational composition.

(b) Let R^n be the composition of R with itself n times for $n \geq 0$. So $R^0 ::= \mathrm{Id}_A$, and $R^{n+1} ::= R \circ R^n$.

Conclude that

$$R^n = W^{(n)} \tag{10.12}$$

for all $n \in \mathbb{N}$.

(c) Conclude that

$$R^+ = \bigcup_{i=1}^{|A|} R^i$$

where R^+ is the positive length walk relation determined by R on the set A.

Problem 10.12.
We can represent a relation S between two sets $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_m\}$ as an $n \times m$ matrix M_S of zeroes and ones, with the elements of M_S defined by the rule

$$M_S(i, j) = 1 \quad \text{IFF} \quad a_i \; S \; b_j.$$

If we represent relations as matrices this way, then we can compute the composition of two relations R and S by a "boolean" matrix multiplication $\otimes$ of their matrices. Boolean matrix multiplication is the same as matrix multiplication except that addition is replaced by OR, multiplication is replaced by AND, and 0 and 1 are used as the Boolean values **False** and **True**. Namely, suppose $R : B \to C$ is a binary relation with $C = \{c_1, \ldots, c_p\}$. So M_R is an $m \times p$ matrix. Then $M_S \otimes M_R$ is an $n \times p$ matrix defined by the rule:

$$[M_S \otimes M_R](i, j) ::= \text{OR}_{k=1}^{m}[M_S(i, k) \text{ AND } M_R(k, j)]. \tag{10.13}$$

Prove that the matrix representation $M_{R \circ S}$ of $R \circ S$ equals $M_S \otimes M_R$ (note the reversal of R and S).

Problem 10.13.
Chickens are rather aggressive birds that tend to establish dominance over other chickens by pecking them—hence the term "pecking order." So for any two chickens in a farmyard, either the first pecks the second, or the second pecks the first, but not both. We say that chicken u *virtually pecks* chicken v if either:

- Chicken u pecks chicken v, or
- Chicken u pecks some other chicken w who in turn pecks chicken v.

A chicken that virtually pecks every other chicken is called a *king chicken*.

We can model this situation with a *chicken digraph* whose vertices are chickens, with an edge from chicken u to chicken v precisely when u pecks v. In the graph

in Figure 10.11, three of the four chickens are kings. Chicken c is not a king in this example since it does not peck chicken b and it does not peck any chicken that pecks chicken b. Chicken a *is* a king since it pecks chicken d, who in turn pecks chickens b and c.

In general, a *tournament digraph* is a digraph with exactly one edge between each pair of distinct vertices.

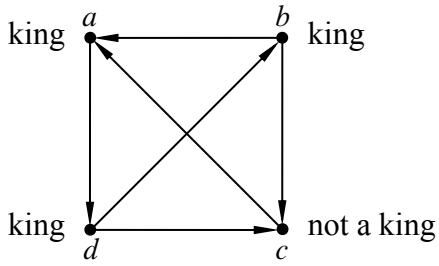


Figure 10.11 A 4-chicken tournament in which chickens a, b and d are kings.

.

(a) Define a 10-chicken tournament graph with a king chicken that has outdegree 1.

(b) Describe a 5-chicken tournament graph in which every player is a king.

(c) Prove

Theorem (King Chicken Theorem)**.** *Any chicken with maximum out-degree in a tournament is a king.*

The King Chicken Theorem means that if the player with the most victories is defeated by another player x, then at least he/she defeats some third player that defeats x. In this sense, the player with the most victories has some sort of bragging rights over every other player. Unfortunately, as Figure 10.11 illustrates, there can be many other players with such bragging rights, even some with fewer victories.

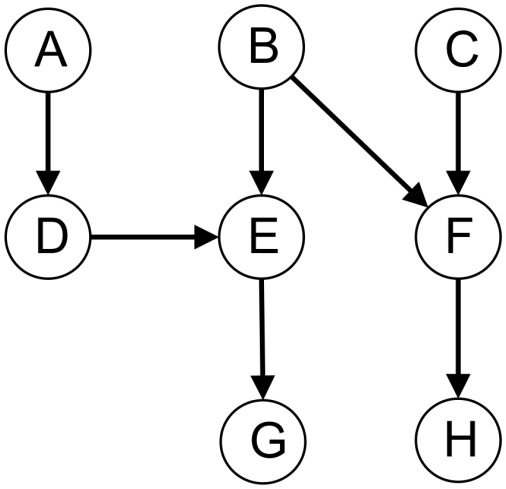
Problems for Section 10.5

Practice Problems

Problem 10.14.

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 10.15.
Let $\{A, ..., H\}$ be a set of tasks that we must complete. The following DAG describes which tasks must be done before others, where there is an arrow from S to T iff S must be done before T.



(a) Write the longest chain.

(b) Write the longest antichain.

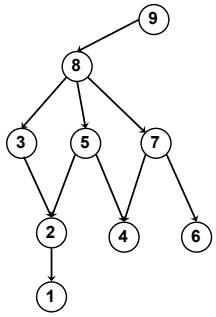
(c) If we allow parallel scheduling, and each task takes 1 minute to complete, what is the minimum amount of time needed to complete all tasks?

Problem 10.16.
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.

Problem 10.17.
Suppose the vertices of a DAG represent tasks taking unit time to complete, and the edges indicate prerequisites among the tasks. Assume there is no bound on how many tasks may be performed in parallel.

What is the smallest number of vertices (tasks) possible in a DAG for which there is more than one minimum time schedule? Carefully justify your answer.

Problem 10.18.
The following DAG describes the prerequisites among tasks $\{1, \ldots, 9\}$.



(a) If each task takes **one hour** to complete, what is the minimum parallel time to complete all the tasks? Briefly explain.

(b) What is the minimum parallel time if no more than two tasks can be completed in parallel? Briefly explain.

Problem 10.19.
The following DAG describes the prerequisites among tasks $\{1, \ldots, 9\}$.

(a) If each task takes unit time to complete, what is the minimum time to complete all the tasks? Briefly explain.

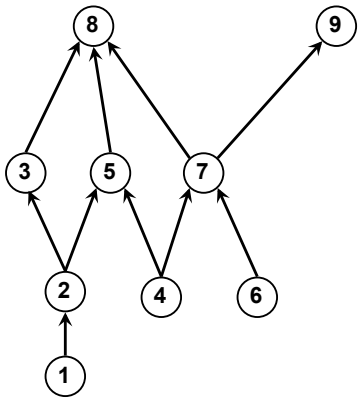
(b) What is the minimum time if no more than two tasks can be completed in parallel? Briefly explain.

Problem 10.20.
Answer the following questions about the dependency DAG shown in Figure 10.12. Assume each node is a task that takes 1 second.

(a) What is the largest chain in this DAG? If there is more than one, only give one.

(b) What is the largest antichain? (Again, give only one if you find there are more than one). Prove there isn't a larger antichain.

(c) How much time would be required to complete all the tasks with a single processor?



(d) How much time would be required to complete all the tasks if there are unlimited processors available.

(e) What is the smallest number of processors that would still allow completion of all the tasks in optimal time? Show a schedule proving it.

Class Problems

Problem 10.21.
The table below lists some prerequisite information for some subjects in the MIT Computer Science program (in 2006). This defines an indirect prerequisite relation that is a DAG with these subjects as vertices.

$18.01 \to 6.042$	$18.01 \to 18.02$
$18.01 \to 18.03$	$6.046 \to 6.840$
$8.01 \to 8.02$	$6.001 \to 6.034$
$6.042 \to 6.046$	$18.03, 8.02 \to 6.002$
$6.001, 6.002 \to 6.003$	$6.001, 6.002 \to 6.004$
$6.004 \to 6.033$	$6.033 \to 6.857$

(a) Explain why exactly six terms are required to finish all these subjects, if you can take as many subjects as you want per term. Using a *greedy* subject selection

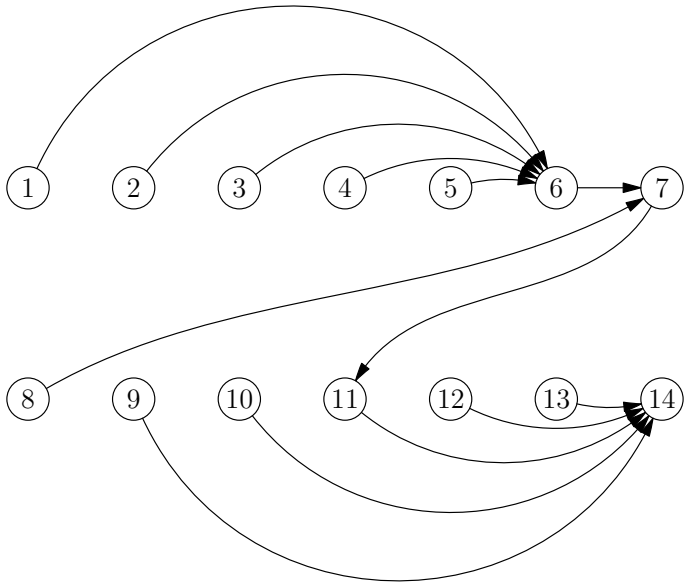


Figure 10.12 Task DAG

strategy, you should take as many subjects as possible each term. Exhibit your complete class schedule each term using a greedy strategy.

(b) In the second term of the greedy schedule, you took five subjects including 18.03. Identify a set of five subjects not including 18.03 such that it would be possible to take them in any one term (using some nongreedy schedule). Can you figure out how many such sets there are?

(c) Exhibit a schedule for taking all the courses—but only one per term.

(d) Suppose that you want to take all of the subjects, but can handle only two per term. Exactly how many terms are required to graduate? Explain why.

(e) What if you could take three subjects per term?

Problem 10.22.

A pair of Math for Computer Science Teaching Assistants, Lisa and Annie, have decided to devote some of their spare time this term to establishing dominion over the entire galaxy. Recognizing this as an ambitious project, they worked out the following table of tasks on the back of Annie's copy of the lecture notes.

1. **Devise a logo** and cool imperial theme music - 8 days.
2. **Build a fleet** of Hyperwarp Stardestroyers out of eating paraphernalia swiped from Lobdell - 18 days.
3. **Seize control** of the United Nations - 9 days, after task #1.
4. **Get shots** for Lisa's cat, Tailspin - 11 days, after task #1.
5. **Open a Starbucks chain** for the army to get their caffeine - 10 days, after task #3.
6. **Train an army** of elite interstellar warriors by dragging people to see *The Phantom Menace* dozens of times - 4 days, after tasks #3, #4, and #5.
7. **Launch the fleet** of Stardestroyers, crush all sentient alien species, and establish a Galactic Empire - 6 days, after tasks #2 and #6.
8. **Defeat Microsoft** - 8 days, after tasks #2 and #6.

We picture this information in Figure 10.13 below by drawing a point for each task, and labelling it with the name and weight of the task. An edge between two points indicates that the task for the higher point must be completed before beginning the task for the lower one.

(a) Give some valid order in which the tasks might be completed.

Lisa and Annie want to complete all these tasks in the shortest possible time. However, they have agreed on some constraining work rules.

- Only one person can be assigned to a particular task; they cannot work together on a single task.
- Once a person is assigned to a task, that person must work exclusively on the assignment until it is completed. So, for example, Lisa cannot work on building a fleet for a few days, run to get shots for Tailspin, and then return to building the fleet.

(b) Lisa and Annie want to know how long conquering the galaxy will take. Annie suggests dividing the total number of days of work by the number of workers, which is two. What lower bound on the time to conquer the galaxy does this give, and why might the actual time required be greater?

(c) Lisa proposes a different method for determining the duration of their project. She suggests looking at the duration of the *critical path*, the most time-consuming sequence of tasks such that each depends on the one before. What lower bound does this give, and why might it also be too low?

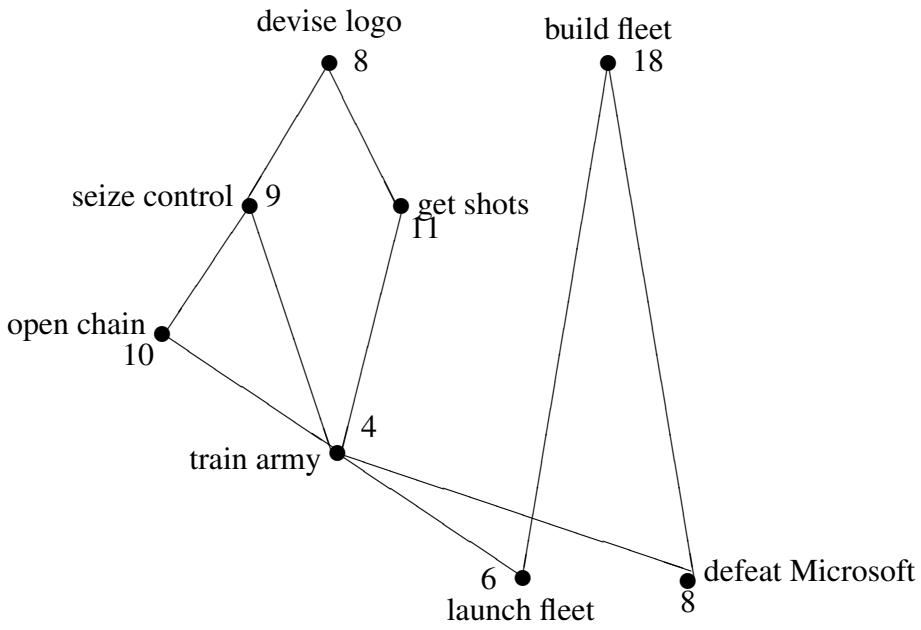


Figure 10.13 Graph representing the task precedence constraints.

(d) What is the minimum number of days that Lisa and Annie need to conquer the galaxy? No proof is required.

Problem 10.23.
Answer the following questions about the powerset pow($\{1, 2, 3, 4\}$) partially ordered by the strict subset relation $\subset$.

(a) Give an example of a maximum length chain.

(b) Give an example of an antchain of size 6.

(c) Describe an example of a topological sort of pow($\{1, 2, 3, 4\}$).

(d) Suppose the partial order describes scheduling constraints on 16 tasks. That is, if

$$A \subset B \subseteq \{1, 2, 3, 4\},$$

then A has to be completed before B starts.[16] What is the minimum number of processors needed to complete all the tasks in minimum parallel time?

[16] As usual, we assume each task requires one time unit to complete.

Prove it.

(e) What is the length of a minimum time **3**-processor schedule?

Prove it.

Homework Problems

Problem 10.24.
The following operations can be applied to any digraph, G:

1. Delete an edge that is in a cycle.
2. Delete edge $\langle u \to v \rangle$ if there is a path from vertex u to vertex v that does not include $\langle u \to v \rangle$.
3. Add edge $\langle u \to v \rangle$ if there is no path in either direction between vertex u and vertex v.

The procedure of repeating these operations until none of them are applicable can be modeled as a state machine. The start state is G, and the states are all possible digraphs with the same vertices as G.

(a) Let G be the graph with vertices $\{1, 2, 3, 4\}$ and edges

$$\{\langle 1 \to 2 \rangle, \langle 2 \to 3 \rangle, \langle 3 \to 4 \rangle, \langle 3 \to 2 \rangle, \langle 1 \to 4 \rangle\}$$

What are the possible final states reachable from G?

A *line graph* is a graph whose edges are all on one path. All the final graphs in part (a) are line graphs.

(b) Prove that if the procedure terminates with a digraph H then H is a line graph with the same vertices as G.

Hint: Show that if H is *not* a line graph, then some operation must be applicable.

(c) Prove that being a DAG is a preserved invariant of the procedure.

(d) Prove that if G is a DAG and the procedure terminates, then the walk relation of the final line graph is a topological sort of G.

Hint: Verify that the predicate

$$P(u, v) ::= \text{there is a directed path from } u \text{ to } v$$

is a preserved invariant of the procedure, for any two vertices u, v of a DAG.

(e) Prove that if G is finite, then the procedure terminates.

Hint: Let s be the number of cycles, e be the number of edges, and p be the number of pairs of vertices with a directed path (in either direction) between them. Note that $p \leq n^2$ where n is the number of vertices of G. Find coefficients a, b, c such that $as + bp + e + c$ is nonnegative integer valued and decreases at each transition.

Problem 10.25.
Let $\prec$ be a strict partial order on a set A and let

$$A_k ::= \{a \mid \text{depth}(a) = k\}$$

where $k \in \mathbb{N}$.

(a) Prove that $A_0, A_1, \ldots$ is a parallel schedule for $\prec$ according to Definition 10.5.7.

(b) Prove that A_k is an antichain.

Problem 10.26.
We want to schedule n tasks with prerequisite constraints among the tasks defined by a DAG.

(a) Explain why any schedule that requires only p processors must take time at least $\lceil n/p \rceil$.

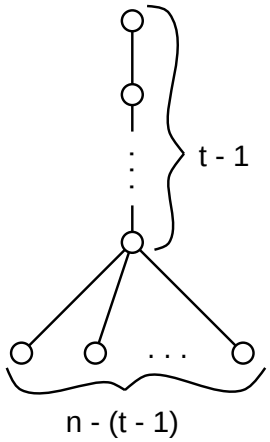
(b) Let $D_{n,t}$ be the DAG with n elements that consists of a chain of $t-1$ elements, with the bottom element in the chain being a prerequisite of all the remaining elements as in the following figure:

What is the minimum time schedule for $D_{n,t}$? Explain why it is unique. How many processors does it require?

(c) Write a simple formula $M(n, t, p)$ for the minimum time of a p-processor schedule to complete $D_{n,t}$.

(d) Show that *every* partial order with n vertices and maximum chain size t has a p-processor schedule that runs in time $M(n, t, p)$.

Hint: Use induction on t.



Problems for Section 10.6

Practice Problems

Problem 10.27.
In this DAG (Figure 10.14) for the divisibility relation on $\{1, \ldots, 12\}$, there is an upward path from a to b iff $a \mid b$. If 24 was added as a vertex, what is the minimum number of edges that must be added to the DAG to represent divisibility on *set* $1, \ldots, 12, 24$? What are those edges?

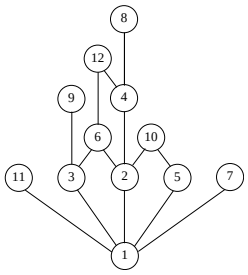


Figure 10.14

Problem 10.28. **(a)** Prove that every strict partial order is a DAG.

(b) What is the smallest possible size of a DAG that is not transitive? Prove it.

(c) Prove that the positive walk relation of a DAG a strict partial order.

Class Problems

Problem 10.29. (a) What are the maxim*al* and minim*al* elements, if any, of the power set pow($\{1, \ldots, n\}$), where n is a positive integer, under the empty relation?

(b) What are the maxim*al* and minim*al* elements, if any, of the set $\mathbb{N}$ of all nonnegative integers under divisibility? Is there a minim*um* or maxim*um* element?

(c) What are the minimal and maximal elements, if any, of the set of integers greater than 1 under divisibility?

(d) Describe a partially ordered set that has no minimal or maximal elements.

(e) Describe a partially ordered set that has a *unique minimal* element, but no minimum element. *Hint:* It will have to be infinite.

Problem 10.30.
The proper subset relation $\subset$ defines a strict partial order on the subsets of [1..6], that is, on pow([1..6]).

(a) What is the size of a maximal chain in this partial order? Describe one.

(b) Describe a largest antichain in this partial order.

(c) What are the maximal and minimal elements? Are they maximum and minimum?

(d) Answer the previous part for the $\subset$ partial order on the set pow $[1..6] - \emptyset$.

Problem 10.31.
If a and b are distinct nodes of a digraph, then a is said to *cover* b if there is an edge from a to b and every path from a to b includes this edge. If a covers b, the edge from a to b is called a *covering edge*.

(a) What are the covering edges in the DAG in Figure 10.15?

(b) Let covering (D) be the subgraph of D consisting of only the covering edges. Suppose D is a finite DAG. Explain why covering (D) has the same positive walk relation as D.

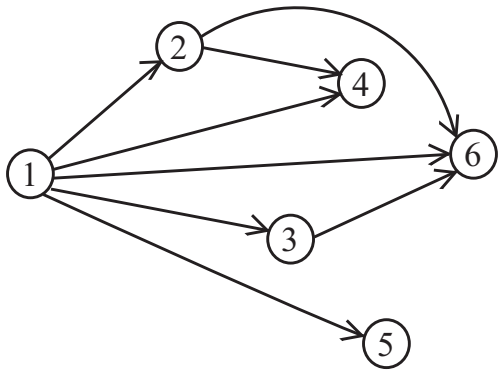


Figure 10.15 DAG with edges not needed in paths

Hint: Consider *longest* paths between a pair of vertices.

(c) Show that if two *DAG*'s have the same positive walk relation, then they have the same set of covering edges.

(d) Conclude that covering (D) is the *unique* DAG with the smallest number of edges among all digraphs with the same positive walk relation as D.

The following examples show that the above results don't work in general for digraphs with cycles.

(e) Describe two graphs with vertices $\{1, 2\}$ which have the same set of covering edges, but not the same positive walk relation (*Hint:* Self-loops.)

(f) (i) The *complete digraph* without self-loops on vertices $1, 2, 3$ has directed edges in each direction between every two distinct vertices. What are its covering edges?

(ii) What are the covering edges of the graph with vertices $1, 2, 3$ and edges $\langle 1 \to 2\rangle$, $\langle 2 \to 3\rangle$, $\langle 3 \to 1\rangle$?

(iii) What about their positive walk relations?

Problems for Section 10.6

Exam Problems

Problem 10.32.
Prove that for any nonempty set D, there is a unique binary relation on D that is both asymmetric and symmetric.

Problem 10.33.
Let D be a set of size $n > 0$. Explain why there are exactly 2^n binary relations on D that are both symmetric and antisymmetric.

Homework Problems

Problem 10.34.
Prove that if R is a transitive binary relation on a set A then $R = R^+$.

Class Problems

Problem 10.35.
Let R be a binary relation on a set D. Each of the following equalities and containments expresses the fact that R has one of the basic relational properties: reflexive, irreflexive, symmetric, asymmetric, antisymmetric, transitive. Identify which property is expressed by each of these formulas and explain your reasoning.

(a) $R \cap \mathrm{Id}_D = \emptyset$

(b) $R \subseteq R^{-1}$

(c) $R = R^{-1}$

(d) $\mathrm{Id}_D \subseteq R$

(e) $R \circ R \subseteq R$

(f) $R \cap R^{-1} = \emptyset$

(g) $R \cap R^{-1} \subseteq \mathrm{Id}_D$

Problems for Section 10.7

Class Problems

Problem 10.36.

Direct Prerequisites	Subject
18.01	6.042
18.01	18.02
18.01	18.03
8.01	8.02
8.01	6.01
6.042	6.046
18.02, 18.03, 8.02, 6.01	6.02
6.01, 6.042	6.006
6.01	6.034
6.02	6.004

(a) For the above table of MIT subject prerequisites, draw a diagram showing the subject numbers with a line going down to every subject from each of its (direct) prerequisites.

(b) Give an example of a collection of sets partially ordered by the proper subset relation $\subset$ that is isomorphic to ("same shape as") the prerequisite relation among MIT subjects from part (a).

(c) Explain why the empty relation is a strict partial order and describe a collection of sets partially ordered by the proper subset relation that is isomorphic to the empty relation on five elements—that is, the relation under which none of the five elements is related to anything.

(d) Describe a *simple* collection of sets partially ordered by the proper subset relation that is isomorphic to the "properly contains" relation $\supset$ on $\text{pow}\{1, 2, 3, 4\}$.

Problem 10.37.

This problem asks for a proof of Lemma 10.7.2 showing that every weak partial order can be represented by (is isomorphic to) a collection of sets partially ordered under set inclusion ($\subseteq$). Namely,

Lemma. *Let $\preceq$ be a weak partial order on a set A. For any element $a \in A$, let*

$$L(a) ::= \{b \in A \mid b \preceq a\},$$
$$\mathcal{L} ::= \{L(a) \mid a \in A\}.$$

Then the function $L : A \to \mathcal{L}$ is an isomorphism from the $\preceq$ relation on A, to the subset relation on $\mathcal{L}$.

(a) Prove that the function $L : A \to \mathcal{L}$ is a bijection.

(b) Complete the proof by showing that

$$a \preceq b \quad \text{iff} \quad \mathrm{L}(a) \subseteq \mathrm{L}(b) \tag{10.14}$$

for all $a, b \in A$.

Homework Problems

Problem 10.38.
Every partial order is isomorphic to a collection of sets under the subset relation (see Section 10.7). In particular, if R is a *strict* partial order on a set A and $a \in A$, define

$$\mathrm{L}(a) ::= \{a\} \cup \{x \in A \mid x \; R \; a\}. \tag{10.15}$$

Then

$$a \; R \; b \quad \text{iff} \quad \mathrm{L}(a) \subset \mathrm{L}(b) \tag{10.16}$$

holds for all $a, b \in A$.

(a) Carefully prove statement (10.16), starting from the definitions of strict partial order and the strict subset relation $\subset$.

(b) Prove that if $\mathrm{L}(a) = \mathrm{L}(b)$ then $a = b$.

(c) Give an example showing that the conclusion of part (b) would not hold if the definition of $\mathrm{L}(a)$ in equation (10.15) had omitted the expression "$\{a\}\cup$."

Problems for Section 10.8

Practice Problems

Problem 10.39.
For each of the binary relations below, state whether it is a strict partial order, a weak partial order, or neither. If it is not a partial order, indicate which of the axioms for partial order it violates.

(a) The superset relation, $\supseteq$ on the power set pow $\{1, 2, 3, 4, 5\}$.

(b) The relation between any two nonnegative integers a, b given by $a \equiv b \pmod 8$.

(c) The relation between propositional formulas G, H given by G IMPLIES H is valid.

(d) The relation 'beats' on Rock, Paper and Scissor (for those who don't know the game "Rock, Paper, Scissors:" Rock beats Scissors, Scissors beats Paper and Paper beats Rock).

(e) The empty relation on the set of real numbers.

(f) The identity relation on the set of integers.

Problem 10.40. (a) Verify that the divisibility relation on the set of nonnegative integers is a weak partial order.

(b) What about the divisibility relation on the set of integers?

Problem 10.41.
Prove directly from the definitions (without appealing to DAG properties) that if a binary relation R on a set A is transitive and irreflexive, then it is asymmetric.

Class Problems

Problem 10.42.
Show that the set of nonnegative integers partially ordered under the divides relation...

(a) ...has a minimum element.

(b) ...has a maximum element.

(c) ...has an infinite chain.

(d) ...has an infinite antichain.

(e) What are the minimal elements of divisibility on the integers greater than 1? What are the maximal elements?

Problem 10.43.
How many binary relations are there on the set $\{0, 1\}$?

How many are there that are transitive?, ... asymmetric?, ... reflexive?, ... irreflexive?, ... strict partial orders?, ... weak partial orders?

Hint: There are easier ways to find these numbers than listing all the relations and checking which properties each one has.

Problem 10.44.
Prove that if R is a partial order, then so is R^{-1}.

Problem 10.45. **(a)** Indicate which of the following relations below are equivalence relations, (**Eq**), strict partial orders (**SPO**), weak partial orders (**WPO**). For the partial orders, also indicate whether it is *linear* (**Lin**).

If a relation is none of the above, indicate whether it is *transitive* (**Tr**), *symmetric* (**Sym**), or *asymmetric* (**Asym**).

(i) The relation $a = b + 1$ between integers a, b,

(ii) The superset relation $\supseteq$ on the power set of the integers.

(iii) The empty relation on the set of rationals.

(iv) The divides relation on the nonegative integers $\mathbb{N}$.

(v) The divides relation on all the integers $\mathbb{Z}$.

(vi) The divides relation on the positive powers of 4.

(vii) The relatively prime relation on the nonnegative integers.

(viii) The relation "has the same prime factors" on the integers.

(b) A set of functions $f, g : D \to \mathbb{R}$ can be partially ordered by the $\leq$ relation, where

$$[f \leq g] ::= \forall d \in D.\, f(d) \leq g(d).$$

Let L be the set of functions $f : \mathbb{R} \to \mathbb{R}$ of the form

$$f(x) = ax + b$$

for constants $a, b \in \mathbb{R}$.

Describe an infinite chain and an infinite anti-chain in L.

Problem 10.46.
In an n-player *round-robin tournament*, every pair of distinct players compete in a single game. Assume that every game has a winner—there are no ties. The results of such a tournament can then be represented with a *tournament digraph* where the vertices correspond to players and there is an edge $\langle x \to y \rangle$ iff x beat y in their game.

(a) Briefly explain why a tournament digraph cannot have cycles of length one or two.

(b) Briefly explain whether the "beats" relation of a tournament graph **always/sometimes/never**...

- ...is asymmetric.
- ...is reflexive.
- ...is irreflexive.
- ...is transitive.

(c) If a tournament graph has no cycles of length three, prove that it is a partial order.

Homework Problems

Problem 10.47.
Let R and S be transitive binary relations on the same set A. Which of the following new relations must also be transitive? For each part, justify your answer with a brief argument if the new relation is transitive or with a counterexample if it is not.

(a) R^{-1}

(b) $R \cap S$

(c) $R \circ R$

(d) $R \circ S$

Problem 10.48.
Digraph G has the *unique path property* when, for any finite nonempty set of vertices of G, there is a unique directed path going through exactly these vertices.

(a) Prove that if G is a linear strict partial order, then G has the unique path property.

(b) Prove conversely that if G has the unique path property, then the positive path relation of G is a linear strict partial order.

Exam Problems

Problem 10.49.
Suppose the precedence constraints on a set of 32 unit time tasks was isomorphic to the powerset, pow($\{1,2,3,4,5\}$) under the strict subset relation $\subset$.

For example, the task corresponding to the set $\{2,4\}$ must be completed before the task corresponding to the set $\{1,2,4\}$ because $\{2,4\} \subset \{1,2,4\}$; the task corresponding to the empty set must be scheduled first because $\emptyset \subset S$ for every nonempty set $S \subseteq \{1,2,3,4,5\}$.

(a) What is the minimum parallel time to complete these tasks?

(b) Describe a maximum size antichain in this partial order.

(c) Briefly explain why the minimum number of processors required to complete these tasks in minimum parallel time is equal to the size of the maximum antichain.

Problem 10.50.
Let R be a weak partial order on a set A. Suppose C is a finite chain.[17]

(a) Prove that C has a maximum element. *Hint:* Induction on the size of C.

(b) Conclude that there is a unique sequence of all the elements of C that is strictly increasing.

Hint: Induction on the size of C, using part (a).

Problems for Section 10.9

Practice Problems

Problem 10.51.
Verify that if *either* of R_1 or R_2 is irreflexive, then so is $R_1 \times R_2$.

Class Problems

Problem 10.52.
Let R_1, R_2 be binary relations on the same set A. A relational property is preserved

[17] A set C is a *chain* when it is nonempty, and all elements $c, d \in C$ are comparable. Elements c and d are *comparable* iff [$c\ R\ d$ OR $d\ R\ c$].

under product, if $R_1 \times R_2$ has the property whenever both R_1 and R_2 have the property.

(a) Verify that each of the following properties are preserved under product.

1. reflexivity,
2. antisymmetry,
3. transitivity.

(b) Verify that if R_1 and R_2 are partial orders and at least one of them is strict, then $R_1 \times R_2$ is a strict partial order.

Problem 10.53.

A partial order on a set A is *well founded* when every non-empty subset of A has a *minimal* element. For example, the less-than relation on a well ordered set of real numbers (see 2.4) is a linear order that is well founded.

Prove that if R and S are well founded partial orders, then so is their product $R \times S$.

Homework Problems

Problem 10.54.

Let S be a sequence of n different numbers. A *subsequence* of S is a sequence that can be obtained by deleting elements of S.

For example, if S is

$$(6, 4, 7, 9, 1, 2, 5, 3, 8),$$

then 647 and 7253 are both subsequences of S (for readability, we have dropped the parentheses and commas in sequences, so 647 abbreviates $(6, 4, 7)$, for example).

An *increasing subsequence* of S is a subsequence of whose successive elements get larger. For example, 1238 is an increasing subsequence of S. Decreasing subsequences are defined similarly; 641 is a decreasing subsequence of S.

(a) List all the maximum-length increasing subsequences of S, and all the maximum-length decreasing subsequences.

Now let A be the *set* of numbers in S. (So A is the integers $[1..9]$ for the example above.) There are two straightforward linear orders for A. The first is numerical order where A is ordered by the $<$ relation. The second is to order the elements by which comes first in S; call this order $<_S$. So for the example above, we would have

$$6 <_S 4 <_S 7 <_S 9 <_S 1 <_S 2 <_S 5 <_S 3 <_S 8$$

Let $\prec$ be the product relation of the linear orders $<_s$ and $<$. That is, $\prec$ is defined by the rule

$$a \prec a' \quad ::= \quad a < a' \text{ AND } a <_S a'.$$

So $\prec$ is a partial order on A (Section 10.9).

(b) Draw a diagram of the partial order $\prec$ on A. What are the maximal and minimal elements?

(c) Explain the connection between increasing and decreasing subsequences of S, and chains and anti-chains under $\prec$.

(d) Prove that every sequence S of length n has an increasing subsequence of length greater than $\sqrt{n}$ or a decreasing subsequence of length at least $\sqrt{n}$.

Problems for Section 10.10

Practice Problems

Problem 10.55.
For each of the following relations, decide whether it is reflexive, whether it is symmetric, whether it is transitive, and whether it is an equivalence relation.

(a) $\{(a, b) \mid a \text{ and } b \text{ are the same age}\}$

(b) $\{(a, b) \mid a \text{ and } b \text{ have the same parents}\}$

(c) $\{(a, b) \mid a \text{ and } b \text{ speak a common language}\}$

Problem 10.56.
For each of the binary relations below, state whether it is a strict partial order, a weak partial order, an equivalence relation, or none of these. If it is a partial order, state whether it is a linear order. If it is none, indicate which of the axioms for partial-order and equivalence relations it violates.

(a) The superset relation $\supseteq$ on the power set pow $\{1, 2, 3, 4, 5\}$.

(b) The relation between any two nonnegative integers a and b such that $a \equiv b \pmod 8$.

(c) The relation between propositional formulas G and H such that [G IMPLIES H] is valid.

(d) The relation between propositional formulas G and H such that $[G \text{ IFF } H]$ is valid.

(e) The relation 'beats' on Rock, Paper, and Scissors (for those who don't know the game Rock, Paper, Scissors, Rock beats Scissors, Scissors beats Paper, and Paper beats Rock).

(f) The empty relation on the set of real numbers.

(g) The identity relation on the set of integers.

(h) The divisibility relation on the integers $\mathbb{Z}$.

Class Problems

Problem 10.57.
Prove Theorem 10.10.4: The equivalence classes of an equivalence relation form a partition of the domain.

Namely, let R be an equivalence relation on a set A and define the equivalence class of an element $a \in A$ to be

$$[a]_R ::= \{b \in A \mid a \ R \ b\}.$$

That is, $[a]_R = R(a)$.

(a) Prove that every block is nonempty and every element of A is in some block.

(b) Prove that if $[a]_R \cap [b]_R \neq \emptyset$, then $a \ R \ b$. Conclude that the sets $[a]_R$ for $a \in A$ are a partition of A.

(c) Prove that $a \ R \ b$ iff $[a]_R = [b]_R$.

Problem 10.58.
For any total function $f : A \to B$ define a relation $\equiv_f$ by the rule:

$$a \equiv_f a' \quad \text{iff} \quad f(a) = f(a'). \tag{10.17}$$

(a) Sketch a proof that $\equiv_f$ is an equivalence relation on A.

(b) Prove that every equivalence relation R on a set A is equal to $\equiv_f$ for the function $f : A \to \text{pow}(A)$ defined as

$$f(a) ::= \{a' \in A \mid a \ R \ a'\}.$$

That is, $f(a) = R(a)$.

Problem 10.59.

Let R be a binary relation on a set D. Each of the following formulas expresses the fact that R has a familiar relational property such as reflexivity, asymmetry, transitivity. Predicate formulas have roman numerals i.,ii.,..., and relational formulas (equalities and containments) are labelled with letters (a),(b),....

Next to each of the relational formulas, write the roman numerals of all the predicate formulas equivalent to it. It is not necessary to name the property expressed, but you can get partial credit if you do. For example, part (a) gets the label "i." It expresses *irreflexivity*.

i. $\forall d.\quad \text{NOT}(d\ R\ d)$

ii. $\forall d.\quad d\ R\ d$

iii. $\forall c, d.\quad c\ R\ d\ \text{IFF}\ d\ R\ c$

iv. $\forall c, d.\quad c\ R\ d\ \text{IMPLIES}\ d\ R\ c$

v. $\forall c, d.\quad c\ R\ d\ \text{IMPLIES NOT}(d\ R\ c)$

vi. $\forall c \neq d.\quad c\ R\ d\ \text{IMPLIES NOT}(d\ R\ c)$

vii. $\forall c \neq d.\quad c\ R\ d\ \text{IFF NOT}(d\ R\ c)$

viii. $\forall b, c, d.\quad (b\ R\ c\ \text{AND}\ c\ R\ d)\ \text{IMPLIES}\ b\ R\ d$

ix. $\forall b, d.\quad [\exists c.\ (b\ R\ c\ \text{AND}\ c\ R\ d)]\ \text{IMPLIES}\ b\ R\ d$

x. $\forall b, d.\quad b\ R\ d\ \text{IMPLIES}\ [\exists c.\ (b\ R\ c\ \text{AND}\ c\ R\ d)]$

(a) $R \cap \text{Id}_D = \emptyset$ ____i.____

(b) $R \subseteq R^{-1}$

(c) $R = R^{-1}$

(d) $\text{Id}_D \subseteq R$

(e) $R \circ R \subseteq R$

(f) $R \subseteq R \circ R$

(g) $R \cap R^{-1} \subseteq \text{Id}_D$

(h) $\overline{R} \subseteq R^{-1}$

(i) $\overline{R} \cap \mathrm{Id}_R = R^{-1} \cap \mathrm{Id}_R$

(j) $R \cap R^{-1} = \emptyset$

Homework Problems

Problem 10.60.
Let R_1 and R_2 be two equivalence relations on a set A. Prove or give a counterexample to the claims that the following are also equivalence relations:

(a) $R_1 \cap R_2$.

(b) $R_1 \cup R_2$.

Problem 10.61.
Say that vertices u, v in a digraph G are *mutually connected* and write

$$u \overset{*}{\longleftrightarrow} v,$$

when there is a path from u to v and also a path from v to u.

(a) Prove that $\overset{*}{\longleftrightarrow}$ is an equivalence relation on $V(G)$.

(b) The blocks of the equivalence relation $\overset{*}{\longleftrightarrow}$ are called the *strongly connected components* of G. Define a relation $\rightsquigarrow$ on the strongly connected components of G by the rule

$$C \rightsquigarrow D \quad \text{IFF there is a path from some vertex in } C \text{ to some vertex in } D.$$

Prove that $\rightsquigarrow$ is a weak partial order on the strongly connected components.

Exam Problems

Problem 10.62.
Let A be a nonempty set.

(a) Describe a single relation on A that is *both* an equivalence relation and a weak partial order on A.

(b) Prove that the relation of part (a) is the only relation on A with these properties.