9 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 9.1. The dots are called *nodes* or *vertices* and the lines are called *directed edges* or *arrows*; the digraph in Figure 9.1 has 4 nodes and 6 directed edges.

Digraphs appear everywhere in computer science. For example, the digraph in Figure 9.2 represents a communication net, a topic we'll explore in depth in Chapter 10. Figure 9.2 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 9.3—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!

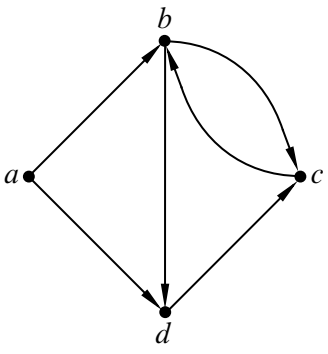

Figure 9.1 A 4-node directed graph with 6 edges.

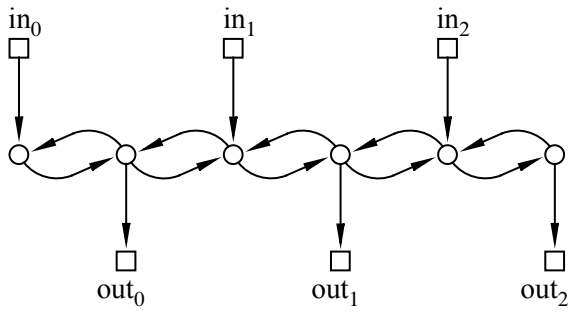

Figure 9.2 A 6-switch packet routing digraph.

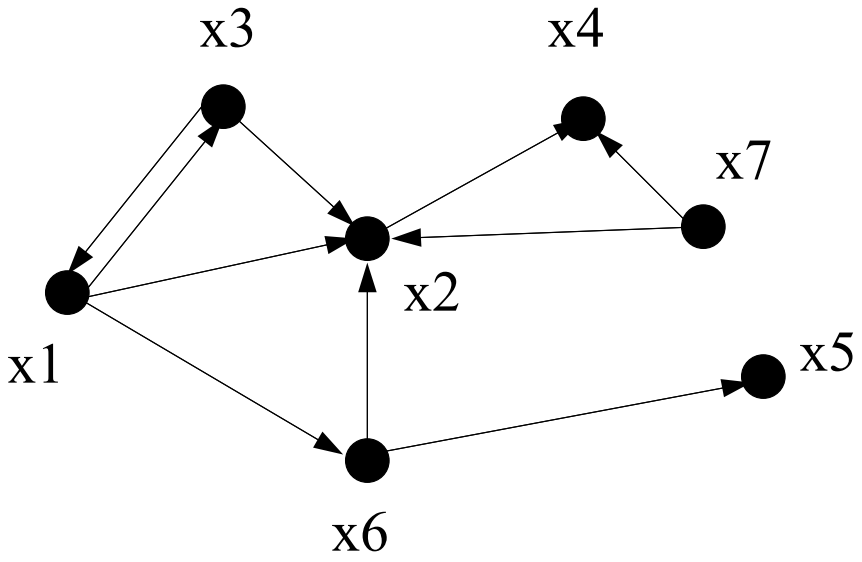

Figure 9.3 Links among Web Pages.

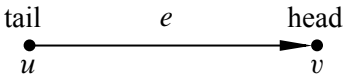

Figure 9.4 A directed edge $e = \langle u \to v \rangle$. The edge e starts at the tail vertex, u, and ends at the head vertex, v.

Definition 9.0.1. 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 9.4. 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 9.0.1 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 9.5.

9.1 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 9.1.1. 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 9.1.2.

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

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

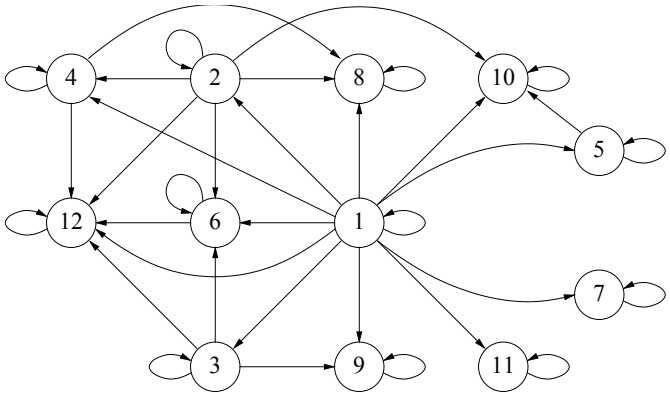

Figure 9.5 The Digraph for Divisibility on $\{1, 2, \ldots, 12\}$.

9.2 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 9.5, 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

$$1 \;\; \langle 1 \to 2 \rangle \;\; 2 \;\; \langle 2 \to 4 \rangle \;\; 4 \;\; \langle 4 \to 12 \rangle \;\; 12 \;\; \langle 12 \to 12 \rangle \;\; 12 \;\; \langle 12 \to 12 \rangle \;\; 12. \tag{9.1}$$

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 9.2.1. A *walk in a digraph*, G, 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, $\mathbf{v}$, 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 9.1 has none, but every vertex in the divisibility relation digraph of Figure 9.5 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 9.1,

- (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 (9.1) through the divisibility graph breaks the walk into the first part of the walk

$$1 \ \langle 1 \to 2 \rangle \ 2 \ \langle 2 \to 4 \rangle \ 4 \tag{9.2}$$

from 1 to 4, and the rest of the walk

$$4\ \langle 4\to 12\rangle\ 12\ \langle 12\to 12\rangle\ 12\ \langle 12\to 12\rangle\ 12. \tag{9.3}$$

from 4 to 12, and we'll say the whole walk (9.1) is the *merge* of the walks (9.2) and (9.3). 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*, $\mathbf{f}^\frown\mathbf{r}$, is the walk that starts with $\mathbf{f}$ and continues with $\mathbf{r}$.[1] Two walks can only be merged if the first ends with the same vertex, v, that the second one starts with. 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 9.2.2.

$$|\mathbf{f}^\frown\mathbf{r}| = |\mathbf{f}| + |\mathbf{r}|.$$

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

9.2.1 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 9.2.3. *The 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 9.2.4. The *distance*, $\text{dist}(u, v)$, 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 9.2.5. *[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 9.2.2, $\mathbf{f}\,\widehat{x}\,\mathbf{r}$ 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 9.2.3.

Proof of the "iff" is in Problem 9.3. ■

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

Lemma 9.2.6. *The shortest positive length closed walk through a vertex is a cycle through that vertex.*

The proof of Lemma 9.2.6 is essentially the same as for Theorem 9.2.3; see Problem 9.7.

9.3 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 9.1. 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 9.3.1. The length-k *walk counting matrix* for an n-vertex graph G is the $n \times n$ matrix C such that

$$C_{uv} ::= \text{the number of length-}k\text{ walks from } u \text{ to } v. \tag{9.4}$$

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 9.3.2. *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 9.3.3. *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 9.3.2. 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,

$$\#\text{length } (k+m) \text{ walks from } u \text{ to } v = \sum_{w \in V(G)} C_{uw} \cdot D_{wv} \tag{9.5}$$

But the right hand side of (9.5) is precisely the definition of $(CD)_{uv}$. Thus, CD is indeed the length-$(k+m)$ walk counting matrix. ■

9.3.1 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 9.3.2 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.

9.4 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

$$u\ G^*\ v ::= \text{there is a walk in } G \text{ from } u \text{ to } v. \tag{9.6}$$

Similarly, there is a *positive walk relation*

$$u\ G^+\ v ::= \text{there is a positive length walk in } G \text{ from } u \text{ to } v. \tag{9.7}$$

Definition 9.4.1. 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.

9.4.1 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 9.4.2. 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

$$a\ (R \circ S)\ c ::= \exists b \in B.\ (a\ S\ b) \text{ AND } (b\ R\ c). \tag{9.8}$$

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 9.9) 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 (9.8) 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]

$$G^* = G^0 \cup G^1 \cup G^2 \cup \ldots \cup G^{|V(G)|-1} = (G \cup G^0)^{|V(G)|-1}. \tag{9.9}$$

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.

---

[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 (9.9) involves a harmless abuse of notation: we should have written

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

6.042J / 18.062J Mathematics for Computer Science
Spring 2015