2 JUL

David: for the edge notation a—b, we need a really long dash to distinguish it from "minus".

5 Graph Theory

Informally, a graph is a bunch of dots with lines connecting some of them. An example is shown in Figure 5.1. The dots are called *nodes* (or *vertices*) and the lines are called *edges*.

and where the lines connect some pairs of dots.

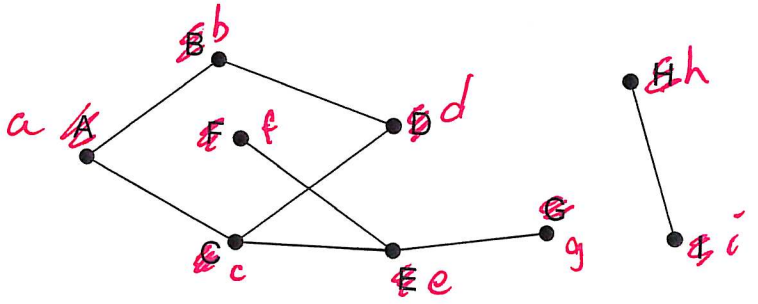


Figure 5.1: An example of a graph with 9 nodes and 8 edges.

Graphs are ubiquitous in computer science because they provide a handy way to represent a relationship between pairs of objects; the objects represent items of interest such as programs, people, cities, or web pages, and we place an edge between a pair of nodes if they are related in a certain way. For example, an edge between a pair of people might indicate that they like (or, in alternate scenarios, that they don't like) each other. An edge between a pair of courses might indicate that one needs to be taken before the other.

In this chapter, we will focus our attention on simple graphs where the relationship denoted by an edge is symmetric. Afterward, in Chapter 6, we consider the situation where the edge denotes a one-way relationship (*e.g.*, where one web page points to the other[1]).

[1]Two Stanford students analyzed such a graph to become multibillionaires. So, pay attention to graph theory, and who knows what might happen!

5.1 Definitions

5.1.1 Simple Graphs

Definition 5.1.1. A *simple graph*, G, consists of a nonempty set, V, called the *vertices* (aka *nodes*[2]) of G, and a set, E, of two-element subsets of V. The members of E are called the *edges* of G, and we write $G = (V, E)$.

The vertices correspond to the dots in Figure 5.1, and the edges correspond to the lines. The graph in Figure 5.1 is expressed mathematically as $G = (V, E)$, where:

$$V = \{A, B, C, D, E, F, G, H, I\}$$

$$E = \{\{A, B\}, \{A, C\}, \{B, D\}, \{C, D\}, \{C, E\}, \{E, F\}, \{E, G\}, \{H, I\}\}.$$

It will often be helpful to use the notation A–B for the edge $\{A, B\}$. Note that A–B and

[2] We will use the terms vertex and node interchangeably.

$B-A$ are different descriptions of the same edge, since sets are unordered. In this case, the graph $G = (V, E)$ has 9 nodes and 8 edges

Definition 5.1.2. Two vertices in a simple graph are said to be *adjacent* if they are joined by an edge, and an edge is said to be *incident* to the vertices it joins. The number of edges incident to a vertex v is called the *degree* of the vertex and is denoted by $\deg(v)$; equivalently, the degree of a vertex is equals the number of vertices adjacent to it.

For example, in the simple graph above, A is adjacent to B and B is adjacent to D, and the edge $A-C$ is incident to vertices A and C. Vertex H has degree 1, D has degree 2, and $\deg(E) = 3$. It is possible for a vertex to have degree 0, in which case it is not adjacent to any other vertices. A simple graph does not need to have any edges at all (in which case, the degree of every vertex is zero and $|E| = 0$)[3], but it does need to have at least one vertex (*i.e.*, $|V| \geq 1$).

[3] Recall that the notation $|E|$ denotes the cardinality of the set E (*i.e.*, the number of elements of E).

Note that simple graphs do *not* have any *self-loops* (*i.e.*, an edge of the form $\{a, a\}$) since an edge is defined to be a set of *two* vertices. In addition, there is at most one edge between any pair of vertices in a simple graph. In other words, a simple graph does not contain *multiedges* or *multiple edges*. That is because E is a set. Lastly, and most importantly, simple graphs do not contain *directed edges* (*i.e.*, edges of the form (a, b) instead of $\{a, b\}$).

There's no harm in relaxing these conditions, and some authors do, but we don't need self-loops, multiple edges between the same two vertices, or graphs with no vertices, and it's simpler not to have them around. We will consider graphs with directed edges (called *directed graphs* or *digraphs*) at length in Chapter 6. Since we'll only be considering simple graphs in this chapter, we'll just call them "graphs" from now on.

5.1.2 Some Common Graphs

Some graphs come up so frequently that they have names. The *complete graph* on n vertices, also called K_n, has an edge between every two vertices. For example, K_5 is shown in Figure 5.2.

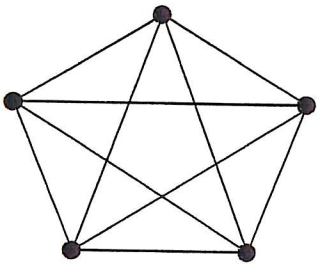


Figure 5.2: The complete graph on 5 nodes, K_5.

The *empty graph* has no edges at all. For example, the empty graph with 5 nodes is shown in Figure 5.3.

The n-node graph containing $n-1$ edges in sequence is known as the *line graph* L_n.

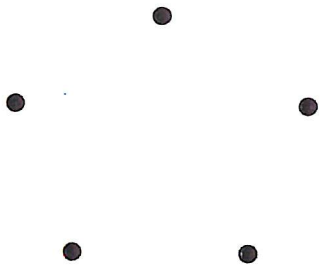


Figure 5.3: The empty graph with 5 nodes.

More formally, $G = (V, E)$ where

$$V = \{v_1, v_2, \ldots, v_n\}$$

and

$$E = \{\, \{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{n-1}, v_n\} \,\}$$

For example, L_5 is displayed in Figure 5.4.

If we add the edge $\{v_n, v_1\}$ to the line graph G, we get the graph C_n consisting of a simple cycle. For example, C_5 is illustrated in Figure 5.5.

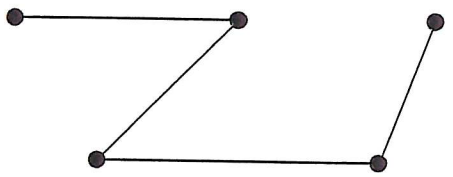


Figure 5.4: The 5-node line graph L_5.

5.1.3 Isomorphism

Two graphs that look the same might actually be different in a formal sense. For example, the two graphs in Figure 5.6 are both simple cycles with 4 vertices, but one graph has vertex set $\{A, B, C, D\}$ while the other has vertex set $\{1, 2, 3, 4\}$. Strictly speaking, these graphs are different mathematical objects, but this is a frustrating distinction since the graphs *look the same*!

Fortunately, we can neatly capture the idea of "looks the same" through the notion of graph isomorphism.

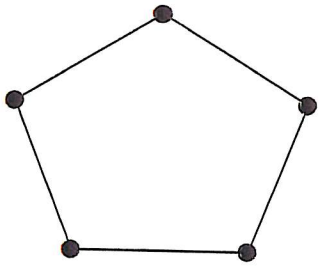


Figure 5.5: The 5-node cycle graph C_5.

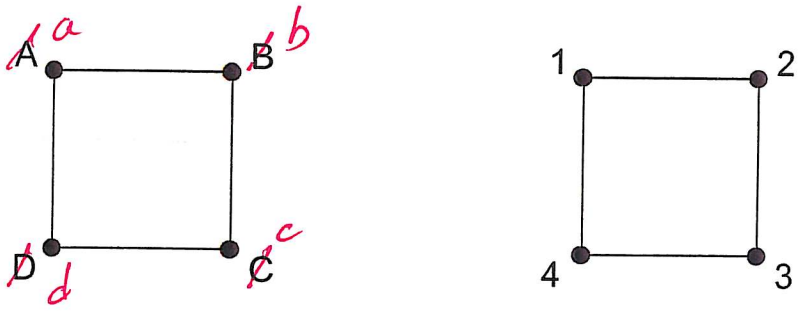
Definition 5.1.3. If $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are two graphs, then we say that G_1 is *isomorphic* to G_2 iff there exists a **bijection**[4] $f : V_1 \to V_2$ such that for every pair of vertices $u, v \in V_1$:

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

The function f is called an *isomorphism* between G_1 and G_2.

In other words, two graphs are isomorphic if they are the same up to a relabeling of

[4] A bijection $f : V_1 \to V_2$ is a function that associates every node in V_1 with a unique node in V_2 and vice-versa. We will study bijections more deeply in Part III.



Redraft graphic: Add (a) and (b) sublabels

Figure 5.6: Two graphs that are isomorphic to C_4.

their vertices. For example, here is an isomorphism between vertices in the two graphs shown in Figure 5.6:

A corresponds to 1
B corresponds to 2
C corresponds to 3.
D corresponds to 4

You can check that there is an edge between two vertices in the graph on the left if and only if there is an edge between the two corresponding vertices in the graph on the right.

Two isomorphic graphs may be drawn very differently. For example, we have shown two different ways of drawing C_5 in Figure 5.7.

Figure 5.7: Two ways of drawing C_5.

Isomorphism preserves the connection properties of a graph, abstracting out what the vertices are called, what they are made out of, or where they appear in a drawing of the graph. More precisely, a property of a graph is said to be *preserved under isomorphism* if whenever G has that property, every graph isomorphic to G also has that property. For example, isomorphic graphs must have the same number of vertices. What's more, if

f is a graph isomorphism that maps a vertex, v, of one graph to the vertex, $f(v)$, of an isomorphic graph, then by definition of isomorphism, every vertex adjacent to v in the first graph will be mapped by f to a vertex adjacent to $f(v)$ in the isomorphic graph. That is, v and $f(v)$ will have the same degree. So if one graph has a vertex of degree 4 and another does not, then they can't be isomorphic. In fact, they can't be isomorphic if the number of degree 4 vertices in each of the graphs is not the same.

Looking for preserved properties can make it easy to determine that two graphs are not isomorphic, or to actually find an isomorphism between them if there is one. In practice, it's frequently easy to decide whether two graphs are isomorphic. However, no one has yet found a *general* procedure for determining whether two graphs are isomorphic that is *guaranteed* to run in polynomial time[5] in $|V|$.

Having an efficient procedure to detect isomorphic graphs would, for example, make

[5] *I.e.*, in an amount of time that is upper-bounded by $|V|^c$ where c is a fixed number independent of $|V|$.

it easy to search for a particular molecule in a database given the molecular bonds. On the other hand, knowing there is no such efficient procedure would also be valuable: secure protocols for encryption and remote authentication can be built on the hypothesis that graph isomorphism is computationally exhausting.

5.1.4 Subgraphs

Definition 5.1.4. A graph $G_1 = (V_1, E_1)$ is said to be a *subgraph* of a graph $G_2 = (V_2, E_2)$ if $V_1 \subseteq V_2$ and $E_1 \subseteq E_2$.

For example, the empty graph on n nodes is a subgraph of C_n, C_n is a subgraph of C_n, and C_n is a subgraph of K_n. Also, the graph $G = (V, E)$ where

$V = \{G, H, I\}$ and $E \in \{\{H, I\}\}$

is a subgraph of the graph in Figure 5.1. On the other hand, any graph containing an

edge $\{G, H\}$ would not be a subgraph of the graph in Figure 5.1 because the graph in Figure 5.1 does not contain this edge.

Note that since a subgraph is itself a graph, the endpoints of any edge in a subgraph must also be in the subgraph. In other words if $G' = (V', E')$ is a subgraph of some graph G, and $\{v_i, v_j\} \in E'$, then it must be the case that $v_i \in V'$ and $v_j \in V'$.

5.1.5 Weighted Graphs

Sometimes, we will use edges to denote a connection between a pair of nodes where the connection has a *capacity* or *weight*. For example, we might be interested in the capacity of an internet fiber between a pair of computers, the resistance of a wire between a pair of terminals, the tension of a spring connecting a pair of devices in a dynamical system, the tension of a bond between a pair of atoms in a molecule, or the distance of a highway between a pair of cities.

redraw with lower case node labels

Redraft graphic: Missing graphic

Figure 5.8: A 4-node weighted graph where the edge A–B has weight 5.

In such cases, it is useful to represent the system with an *edge-weighted* graph (aka a *weighted graph*). A weighted graph is the same as a simple graph except that we associate a real number (*i.e.*, the weight) with each edge in the graph. Mathematically speaking, a weighted graph consists of a graph $G = (V, E)$ and a weight function $w : E \to \mathbb{R}$. For example, Figure 5.8 shows a weighted graph where the weight of edge A–B is 5.

5.1.6 Adjacency Matrices

There are many ways to represent a graph. We have already seen two ways: you can draw it (*e.g.*, as in Figure 5.8), and you can represent it with sets (as in $G = (V, E)$). Another common representation is with an adjacency matrix.

Definition 5.1.5. Given an n-node graph $G = (V, E)$ where $V = \{v_1, v_2, \ldots, v_n\}$, the *adjacency matrix* for G is the $n \times n$ matrix $A_G = \{a_{ij}\}$ where

$$a_{ij} = \begin{cases} 1 & \text{if } \{v_i, v_j\} \in E \\ 0 & \text{otherwise.} \end{cases}$$

If G is a weighted graph with edge weight given by $w : E \to \mathbb{R}$, then the adjacency matrix for G is $A_G = \{a_{ij}\}$ where

$$a_{ij} = \begin{cases} w(\{v_i, v_j\}) & \text{if } \{v_i, v_j\} \in E \\ 0 & \text{otherwise.} \end{cases}$$

For example, Figure 5.9 displays the adjacency matrices for the graphs shown in Figures 5.6(a) and 5.8 where $v_1 = A$, $v_2 = B$, $v_3 = C$, and $v_4 = D$.

$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 5 & 0 & 0 \\ 5 & 0 & 6 & 0 \\ 0 & 6 & 0 & -3 \\ 0 & 0 & -3 & 0 \end{pmatrix}$$

(a) (b)

Figure 5.9: Examples of adjacency matrices. (a) shows the adjacency matrix for the graph in Figure 5.6(a) and (b) shows the adjacency matrix for the weighted graph in Figure 5.8. In each case, we use the set $v_1 = A$, $v_2 = B$, $v_3 = C$, and $v_4 = D$ to construct the matrix.

5.2 Matching Problems

We begin our study of graph theory by considering the scenario where the nodes in a graph represent people and edges represent a relationship between pairs of people such as "likes", "marries", and so on. Now, you may be wondering what marriage has

to do with computer science, and with good reason. It turns out that the techniques we will develop apply to much more general scenarios where instead of matching men to women, we need to match packets to paths in a network, applicants to jobs, or internet traffic to web servers. And, as we will describe later, these techniques are widely used in practice.

In our first example, we will show how graph theory can be used to debunk an urban legend about sexual practices in America. Yes, you read correctly. So, fasten your seat belt—who knew that math might actually be interesting!

5.2.1 Sex in America

On average, who has more opposite-gender partners: men or women?

Sexual demographics have been the subject of many studies. In one of the largest, researchers from the University of Chicago interviewed a random sample of 2500 Americans over several years to try to get an answer to this question. Their study, published

in 1994, and entitled *The Social Organization of Sexuality* found that on average men have 74% more opposite-gender partners than women.

Other studies have found that the disparity is even larger. In particular, ABC News claimed that the average man has 20 partners over his lifetime, and the average woman has 6, for a percentage disparity of 233%. The ABC News study, aired on Primetime Live in 2004, purported to be one of the most scientific ever done, with only a 2.5% margin of error. It was called "American Sex Survey: A peek between the sheets." The promotion for the study is even better:

> "A ground breaking ABC News 'Primetime Live' survey finds a range of eye-popping sexual activities, fantasies and attitudes in this country, confirming some conventional wisdom, exploding some myths—and venturing where few scientific surveys have gone before."

Probably that last part about going where few scientific surveys have gone before is pretty accurate!

Yet again, in August, 2007, the N.Y. Times reported on a study by the National Center for Health Statistics of the U.S. Government showing that men had seven partners while women had four.

Anyway, whose numbers do you think are more accurate, the University of Chicago, ABC News, or the National Center for Health Statistics?—don't answer; this is a setup question like "When did you stop beating your wife?" Using a little graph theory, we will now explain why none of these findings can be anywhere near the truth.

Let's model the question of heterosexual partners in graph theoretic terms. To do this, we'll let G be the graph whose vertices, V, are all the people in America. Then we split V into two separate subsets: M, which contains all the males, and F, which contains all

the females.[6] We'll put an edge between a male and a female iff they have been sexual partners. A possible subgraph of this graph is illustrated in Figure 5.10 with males on the left and females on the right.

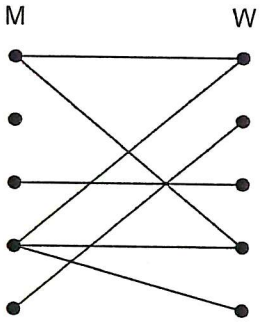


Figure 5.10: The sex partners graph

Actually, G is a pretty hard graph to figure out, let alone draw. The graph is *enormous*: the US population is about 300 million, so $|V| \approx 300M$. In the United States, approximately 50.8% of the populatin is female and 49.2% is male, and so $|M| \approx 147.6M$, and

[6] For simplicity, we'll ignore the possibility of someone being both, or neither, a man and a woman.

$|F| \approx 152.4M$. And we don't even have trustworthy estimates of how many edges there are, let alone exactly which couples are adjacent. But it turns out that we don't need to know any of this to debunk the sex surveys—we just need to figure out the relationship between the average number of partners per male and partners per female. To do this, we note that every edge is incident to exactly one M vertex and one F vertex (remember, we're only considering male-female relationships); so the sum of the degrees of the M vertices equals the number of edges, and the sum of the degrees of the F vertices equals the number of edges. So these sums are equal:

$$\sum_{x \in M} \deg(x) = \sum_{y \in F} \deg(y).$$

If we divide both sides of this equation by the product of the sizes of the two sets, $|M| \cdot |F|$, we obtain

$$\left(\frac{\sum_{x \in M} \deg(x)}{|M|}\right) \cdot \frac{1}{|F|} = \left(\frac{\sum_{y \in F} \deg(y)}{|F|}\right) \cdot \frac{1}{|M|} \tag{5.1}$$

Notice that

$$\frac{\sum_{x \in M} \deg(x)}{|M|}$$

is simply the average degree of a node in M. This is the average number of opposite-gender partners for a male in America. Similarly,

$$\frac{\sum_{x \in F} \deg(x)}{|F|}$$

is the average degree of a node in F, which is the average number of opposite-gender partners for a female in America. Hence, Equation 5.1 implies that on average, an American male has $|F|/|M|$ times as many opposite-gender partners as the average American female.

From the Census Bureau reports, we know that there are slightly more females than males in America; in particular $|F| / |M|$ is about 1.035. So we know that on average, males have 3.5% more opposite-gender partners than females. Of course, this statistic

really says nothing about any sex's promiscuity or selectivity. Remarkably, promiscuity is completely irrelevant in this analysis. That is because the ratio of the average number of partners is completely determined by the relative number of males and females. Collectively, males and females have the same number of opposite gender partners, since it takes one of each set for every partnership, but there are fewer males, so they have a higher ratio. This means that the University of Chicago, ABC, and the Federal Government studies are way off. After a huge effort, they gave a totally wrong answer.

There's no definite explanation for why such surveys are consistently wrong. One hypothesis is that males exaggerate their number of partners—or maybe females downplay theirs—but these explanations are speculative. Interestingly, the principal author of the National Center for Health Statistics study reported that she knew the results had to be wrong, but that was the data collected, and her job was to report it.

The same underlying issue has led to serious misinterpretations of other survey data.

For example, a few years ago, the Boston Globe ran a story on a survey of the study habits of students on Boston area campuses. Their survey showed that on average, minority students tended to study with non-minority students more than the other way around. They went on at great length to explain why this "remarkable phenomenon" might be true. But it's not remarkable at all—using our graph theory formulation, we can see that all it says is that there are fewer minority students than non-minority students, which is, of course what "minority" means.

The Handshaking Lemma

The previous argument hinged on the connection between a sum of degrees and the number edges. There is a simple connection between these quantities in any graph:

Lemma 5.2.1. *The sum of the degrees of the vertices in a graph equals twice the number of edges.*

Proof. Every edge contributes two to the sum of the degrees, one for each of its endpoints. ■

Lemma 5.2.1 is sometimes called the *Handshake Lemma*: if we total up the number of people each person at a party shakes hands with, the total will be twice the number of handshakes that occurred.

5.2.2 Bipartite Matchings

Bipartite Graphs

There were two kinds of vertices in the "Sex in America" graph—males and females, and edges only went between the two kinds. Graphs like this come up so frequently that they have earned a special name—they are called *bipartite graphs*.

Definition 5.2.2. A *bipartite graph* is a graph together with a partition of its vertices into

two sets, L and R, such that every edge is incident to a vertex in L and to a vertex in R.

So every bipartite graph looks something like the graph in Figure 5.10.

The Problem

The bipartite matching problem is related to the sex-in-America problem that we just studied; only now the goal is to get everyone happily married, or at least to pair up every man with a woman that he likes, and vice-versa. As you might imagine, this is not possible for a variety of reasons, not the least of which is the fact that there are more women in America than men. So, it is simply not possible to pair up every woman with a man so that every man is paired with at most one woman.

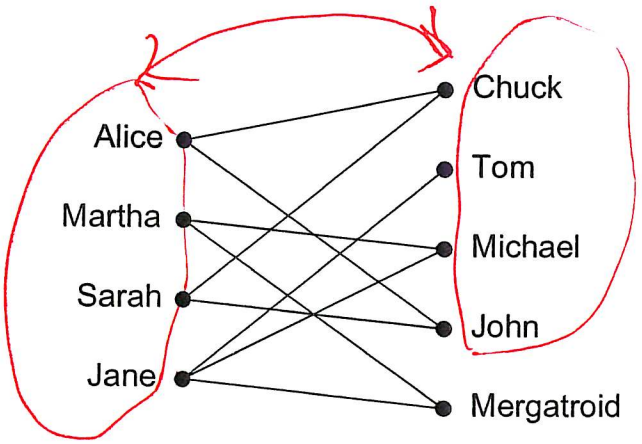
But what about getting a mate for every man so that every woman is paired with at most one man? Is it possible to do this so that each man is paired with a woman that he likes? The answer, of course, depends on the bipartite graph that represents

who likes who, but the good news is that it is possible to find natural properties of the who-likes-who graph that completely determine the answer to this question.

In general, suppose that we have a set of men and an equal-sized or larger set of women, and there is a graph with an edge between a man and a woman if the man likes the woman. (Note that in this scenario, the "likes" relationship need no be symmetric, since for the time being, we will only worry about finding a mate for man that he likes.[7] Later, we will consider the "likes" relationship from the female perspective as well.) For example, we might obtain the graph in Figure 5.11.

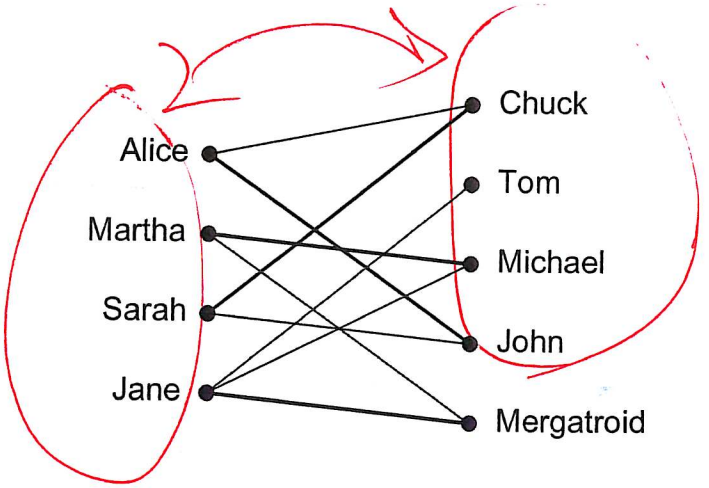
In this problem, a *matching* will mean a way of assigning every man to a woman so that different men are assigned to different women, and a man is always assigned to

[7] By the way, we do not mean to imply that males or pairs should or should not be of a heterosexual nature. Nor do we mean to imply that men should get their choice instead of women. It's just that with bipartite graphs, the edges only connected male nodes to female nodes and there are fewer men in America. So please don't read to much into this example.



Redraft graphic: Change labels

Figure 5.11: A graph where an edge between a man and woman denotes that the man likes the woman.



Redraft graphic: Redo labels

Figure 5.12: One possible matching for the men is shown with bold edges. For example, John is matched with Jane.

a woman that he likes. For example, one possible matching for the men is shown in Figure 5.12.

The Matching Condition

A famous result known as Hall's Matching Theorem gives necessary and sufficient conditions for the existence of a matching in a bipartite graph. It turns out to be a remark-

ably useful mathematical tool.

We'll state and prove Hall's Theorem using man-likes-woman terminology. Define *the set of women liked by a given set of men* to consist of all women liked by at least one of those men. For example, the set of women liked by Tom and John consists of Martha, Sarah, and Mergatroid. For us to have any chance at all of matching up the men, the following *matching condition* must hold:

Every subset of men likes at least as large a set of women.

For example, we can not find a matching if some set of 4 men like only 3 women. Hall's Theorem says that this necessary condition is actually sufficient; if the matching condition holds, then a matching exists.

Theorem 5.2.3. *A matching for a set of men M with a set of women W can be found if and only if the matching condition holds.*

Proof. First, let's suppose that a matching exists and show that the matching condition holds. Consider an arbitrary subset of men. Each man likes at least the woman he is matched with. Therefore, every subset of men likes at least as large a set of women. Thus, the matching condition holds.

Next, let's suppose that the matching condition holds and show that a matching exists. We use strong induction on $|M|$, the number of men,

Base Case: ($|M| = 1$) If $|M| = 1$, then the matching condition implies that the lone man likes at least one woman, and so a matching exists.

Inductive Step: Now suppose that $|M| \geq 2$. There are two cases:

Case 1: Every proper subset[8] of men likes a *strictly larger* set of women. In this case, we have some latitude: we pair an arbitrary man with a woman he likes and send them both away. The matching condition still holds for the remaining men and

[8] Recall that a subset A of B is *proper* if $A \neq B$.

women since we have removed only one woman, so we can match the rest of the men by induction.

Case 2: Some proper subset of men $X \subset M$ likes an *equal-size* set of women $Y \subset W$. We match the men in X with the women in Y by induction and send them all away. We can also match the rest of the men by induction if we show that the matching condition holds for the remaining men and women. To check the matching condition for the remaining people, consider an arbitrary subset of the remaining men $X' \subseteq (M - X)$, and let Y' be the set of remaining women that they like. We must show that $|X'| \leq |Y'|$. Originally, the combined set of men $X \cup X'$ liked the set of women $Y \cup Y'$. So, by the matching condition, we know:

$$|X \cup X'| \leq |Y \cup Y'|$$

We sent away $|X|$ men from the set on the left (leaving X') and sent away an equal

Definition 5.2.4. A *matching* in a graph, G, is a set of edges such that no two edges in the set share a vertex. A matching is said to *cover* a set, L, of vertices iff each vertex in L has an edge of the matching incident to it. A matching is said to be *perfect* if every node in the graph is incident to an edge in the matching. In any graph, the set $N(S)$, of *neighbors* of some set, S, of vertices is the set of all vertices adjacent to some vertex in S. That is,

$$N(S) ::= \{\, r \mid s\text{–}r \text{ is an edge for some } s \in S \,\}.$$

S is called a *bottleneck* if

$$|S| > |N(S)|.$$

Theorem 5.2.5 (Hall's Theorem). *Let G be a bipartite graph with vertex partition L, R. There is matching in G that covers L iff no subset of L is a bottleneck.*

An Easy Matching Condition

The bipartite matching condition requires that *every* subset of men has a certain property. In general, verifying that every subset has some property, even if it's easy to check any particular subset for the property, quickly becomes overwhelming because the number of subsets of even relatively small sets is enormous—over a billion subsets for a set of size 30. However, there is a simple property of vertex degrees in a bipartite graph that guarantees the existence of a matching. Namely, call a bipartite graph *degree-constrained* if vertex degrees on the left are at least as large as those on the right. More precisely,

Definition 5.2.6. A bipartite graph G with vertex partition L, R where $|L| \leq |R|$ is *degree-constrained* if $\deg(l) \geq \deg(r)$ for every $l \in L$ and $r \in R$.

For example, the graph in Figure 5.27 is degree constrained since every node on the

left is adjacent to at least two nodes on the right while every node on the right is incident to at most two nodes on the left.

Regular graphs provide a large class of graphs that often arise in practice that are degree constrained.

Definition 5.2.7. A graph is said to be *regular* if every node has the same degree.

The following lemma implies that every degree-constrained bipartite graph has a matching.

Lemma 5.2.8. *Let G be a bipartite graph with vertex partition L, R where $|L| \leq |R|$. If G is degree-constrained, then no subset of L is a bottleneck.*

Proof. Let S be any set of vertices in L. The number of edges incident to vertices in S is exactly the sum of the degrees of the vertices in S. Each of these edges is incident to a vertex in $N(S)$ by definition of $N(S)$. So the sum of the degrees of the vertices in $N(S)$

INSERT 6A

Proof: The proof is by contradiction. Suppose that G is degree constrained by that there is no matching that covers L. By Theorem 5.2.5, this means that there must be a bottleneck $S \subseteq L$.

Let d be a value such that $\deg(l) \geq x \geq \deg(r)$ for every $l \in L$ and $r \in R$. Since every edge incident to a node in S is incident to a node in $N(S)$, we know that

$$|N(S)|x \geq |S|x$$

and thus that

$$|N(S)| \geq |S|.$$

This means that S is not a bottleneck, which is a contradiction. Hence G has a matching that covers L. ∎

is at least as large as the sum for S. But since the degree of every vertex in $N(S)$ is at most as large as the degree of every vertex in S, there would have to be at least as many terms in the sum for $N(S)$ as in the sum for S. So there have to be at least as many vertices in $N(S)$ as in S, proving that S is not a bottleneck. ■

Corollary 5.2.9. *Let G be a bipartite graph with vertex partition L, R where $|L| \leq |R|$. If G is degree-constrained, then there is a matching that covers L.*

Proof. Combine Lemma 5.2.8 with Theorem 5.2.5. ■

We can now prove:

INSERT GB goes here
text from p361

Theorem 5.2.10. *Every regular bipartite graph has a perfect matching.*

Proof. Let G be a regular bipartite graph with vertex partition L, R where $|L| \leq |R|$. Since regular graphs are degree-constrained, we know by Corollary 5.2.9 that there must be a matching in G that covers L. Since G is regular, we also know that $|L| = |R|$ and

Theorem 5.2.8

thus the matching must also cover R. This means that every node in G is incident to an edge in the matching and thus G has a perfect matching. ■

Theorem 5.2.10 is a surprisingly useful tool in graph theory. For example, we will use it in Chapter 6 when we are figuring out how to route data paths in a Benês network.

5.2.3 The Stable Marriage Problem

The Problem

We next consider a version of the bipartite matching problem where there are an equal number of men and women, and where each person has preferences about who they would like to marry. In fact, we assume that each man he has a complete list of all the women ranked according to his preferences, with no ties. Likewise, each woman has a ranked list of all of the men.

The preferences don't have to be symmetric. That is, Jennifer might like Brad best, but Brad doesn't necessarily like Jennifer best. The goal is to marry everyone: every man must marry exactly one woman and vice-versa—no polygamy. Moreover, we would like to find a matching between men and women that is *stable* in the sense that there is no pair of people that prefer each other to their spouses.

For example, suppose *every* man likes Angelina best, and every woman likes Brad best, but Brad and Angelina are married to other people, say Jennifer and Billy Bob. Now *Brad and Angelina prefer each other to their spouses*, which puts their marriages at risk: pretty soon, they're likely to start spending late nights together working on problem sets!

This unfortunate situation is illustrated in Figure 5.13, where the digits "1" and "2" near a man shows which of the two women he ranks first second, respectively, and similarly for the women.

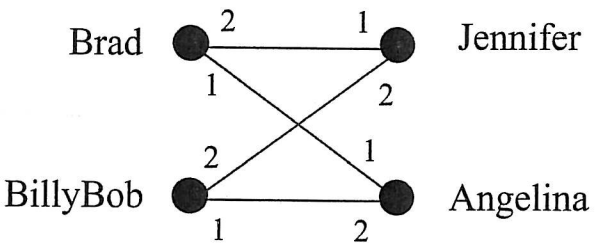


Figure 5.13: Preferences for four people. Both men like Angelina best and both women like Brad best.

More generally, in any matching, a man and woman who are not married to each other and who like each other better than their spouses, is called a *rogue couple.* In the situation shown in Figure 5.13, Brad and Angelina would be a rogue couple.

Having a rogue couple is not a good thing, since it threatens the stability of the marriages. On the other hand, if there are no rogue couples, then for any man and woman who are not married to each other, at least one likes their spouse better than the other, and so they won't be tempted to start an affair.

Definition 5.2.11. A *stable matching* is a matching with no rogue couples.

The question is, given everybody's preferences, how do you find a stable set of marriages? In the example consisting solely of the four people in Figure 5.13, we could let Brad and Angelina both have their first choices by marrying each other. Now neither Brad nor Angelina prefers anybody else to their spouse, so neither will be in a rogue couple. This leaves Jen not-so-happily married to Billy Bob, but neither Jen nor Billy Bob can entice somebody else to marry them, and so there is a stable matching.

Surprisingly, It is something of a surprise that there always is a stable matching among a group of men and women, and we'll shortly explain why. The surprise springs in part from considering the apparently similar "buddy" matching problem. That is, if people can be paired off as buddies, regardless of gender, then a stable matching *may not* be possible. For example, Figure 5.14 shows a situation with a love triangle and a fourth person who

is everyone's last choice. In this figure Mergatoid's preferences aren't shown because they don't even matter.

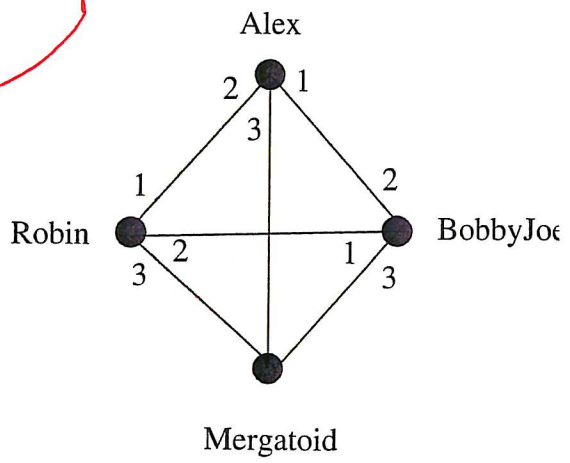


Figure 5.14: Some preferences with no stable buddy matching.

Let's see why there is no stable matching:

Lemma 5.2.12. *There is no stable buddy matching among the four people in Figure 5.14.*

Proof. We'll prove this by contradiction.

Assume, for the purposes of contradiction, that there is a stable matching. Then there

are two members of the love triangle that are matched. Since preferences in the triangle are symmetric, we may assume in particular, that Robin and Alex are matched. Then the other pair must be Bobby-Joe matched with Mergatoid.

But then there is a rogue couple: Alex likes Bobby-Joe best, and Bobby-Joe prefers Alex to his buddy Mergatoid. That is, Alex and Bobby-Joe are a rogue couple, contradicting the assumed stability of the matching. ■

So getting a stable *buddy* matching may not only be hard, it may be impossible. But when mens are only allowed to marry women, and vice versa, then it turns out that a stable matching can always be found.[9]

[9]Once again, we disclaim any political statement here—its just the way that the math works out.

The Mating Ritual

The procedure for finding a stable matching involves a *Mating Ritual* that takes place over several days. The following events happen each day:

Morning: Each woman stands on her balcony. Each man stands under the balcony of his favorite among the women on his list, and he serenades her. If a man has no women left on his list, he stays home and does his math homework.

Afternoon: Each woman who has one or more suitors serenading her, says to her favorite among them, "We might get engaged. Come back tomorrow." To the other suitors, she says, "No. I will never marry you! Take a hike!"

Evening: Any man who is told by a woman to take a hike, crosses that woman off his list.

Termination condition: When a day arrives in which every woman has at most one

suitor, the ritual ends with each woman marrying her suitor, if she has one.

There are a number of facts about this Mating Ritual that we would like to prove:

- The Ritual eventually reaches the termination condition.
- Everybody ends up married.
- The resulting marriages are stable.

There is a Marriage Day

It's easy to see why the Mating Ritual has a terminal day when people finally get married. Every day on which the ritual hasn't terminated, at least one man crosses a woman off his list. (If the ritual hasn't terminated, there must be some woman serenaded by at least two men, and at least one of them will have to cross her off his list). If we start with n men and n women, then each of the n men's lists initially has n women on it, for

a total of n^2 list entries. Since no women ever gets added to a list, the total number of entries on the lists decreases every day that the Ritual continues, and so the Ritual can continue for at most n^2 days.

They All Live Happily Every After...

We still have to prove that the Mating Ritual leaves everyone in a stable marriage. To do this, we note one very useful fact about the Ritual: if a woman has a favorite suitor on some morning of the Ritual, then that favorite suitor will still be serenading her the next morning—because his list won't have changed. So she is sure to have today's favorite man among her suitors tomorrow. That means she will be able to choose a favorite suitor tomorrow who is at least as desirable to her as today's favorite. So day by day, her favorite suitor can stay the same or get better, never worse. This sounds like an invariant, and it is.

Definition 5.2.13. Let P be the predicate: For every woman, w, and every man, m, if w is crossed off m's list, then w has a suitor whom she prefers over m.

Lemma 5.2.14. *P is an invariant for The Mating Ritual.*

Proof. By induction on the number of days.

Base Case: In the beginning (*i.e.*, at the end of day 0), every woman is on every list—no one has been crossed off and so P is vacuously true.

Inductive Step: Assume P is true on day d and let w be a woman that has been crossed off a man m's list by the end of day $d+1$.

Case 1: w was crossed off m's list on day $d+1$. Then, w must have a suitor she prefers on day $d+1$.

Case 2: w was crossed off m's list prior to day $d+1$. Since P is true at the end of day d, this means that w has a suitor she prefers to m on day d. She therefore has the

same suitor or someone she prefers better at the end of day $d + 1$.

In both cases, P is true at the end of day $d + 1$ and so P must be an invariant. ■

With Lemma 5.2.14 in hand, we can now prove:

Theorem 5.2.15. *Everyone is married by the Mating Ritual.*

Proof. By contradiction. Assume that it is the last day of the Mating Ritual and someone does not get married. Since there are an equal number of men and women, and since bigamy is not allowed, this means that at least one man (call him Bob) and at least one woman do not get married.

Since Bob is not married, he can't be serenading anybody and so his list must be empty. So by invariant P, every woman has a suitor whom she prefers to Bob. Since it is the last day and every woman still has a suitor, this means that every woman gets married. This is a contradiction since we already argued that at least one woman is *not*

married. Hence our assumption must be false and so everyone must be married. ■

Theorem 5.2.16. *The Mating Ritual produces a stable matching.*

Proof. Let Brad and Jen be any man and woman, respectively, that are *not* married to each other on the last day of the Mating Ritual. We will prove that Brad and Jen are not a rogue couple, and thus that all marriages on the last day are stable. There are two cases to consider.

Case 1: Jen is not on Brad's list by the end. Then by invariant P, we know that Jen has a suitor (and hence a husband) that she prefers to Brad. So she's not going to run off with Brad—Brad and Jen cannot be a rogue couple.

Case 2: Jen is on Brad's list. But since Brad is not married to Jen, he must be choosing to serenade his wife instead of Jen, so he must prefer his wife. So he's not going to run off with Jen—once again, Brad and Jenn are not a rogue couple. ■

...Especially the Men

Who is favored by the Mating Ritual, the men or the women? The women *seem* to have all the power: they stand on their balconies choosing the finest among their suitors and spurning the rest. What's more, we know their suitors can only change for the better as the Ritual progresses. Similarly, a man keeps serenading the woman he most prefers among those on his list until he must cross her off, at which point he serenades the next most preferred woman on his list. So from the man's perspective, the woman he is serenading can only change for the worse. Sounds like a good deal for the woman.

But it's not! The fact is that from the beginning, the men are serenading their first choice woman, and the desirability of the woman being serenaded decreases only enough to ~~give the man his most desirable possible spouse~~ insure overall stability. The Mating Ritual actually does as well as possible for all the men and does the worst possible job for the women.

their realm of possibility.

Everybody has an optimal and a pessimal spouse, since we know there is at least one stable matching, namely the one produced by the Mating Ritual. Now here is the shocking truth about the Mating Ritual:

Theorem 5.2.18. *The Mating Ritual marries every man to his optimal spouse.*

Proof. By contradiction. Assume for the purpose of contradiction that some man does not get his optimal spouse. Then there must have been a day when he crossed off his optimal spouse—otherwise he would still be serenading (and would ultimately marry) her or some even more desirable woman.

By the Well Ordering Principle, there must be a *first* day when a man (call him "Keith") crosses off his optimal spouse (call her Nicole).

According to the rules of the Ritual, Keith crosses off Nicole because Nicole has a

preferred suitor (call him Tom), so

Nicole prefers Tom to Keith. (*)

Since this is the first day an optimal woman gets crossed off, we know that Tom had not previously crossed off his optimal spouse, and so

Tom ranks Nicole at least as high as his optimal spouse. (**)

By the definition of an optimal spouse, there must be some stable set of marriages in which Keith gets his optimal spouse, Nicole. But then the preferences given in (*) and (**) imply that Nicole and Tom are a rogue couple within this supposedly stable set of marriages (think about it). This is a contradiction. ■

Theorem 5.2.19. *The Mating Ritual marries every woman to her pessimal spouse.*

Proof. By contradiction. Assume that the theorem is not true. Hence there must be a stable set of marriages $\mathcal{M}$ where some woman (call her Nicole) is married to a man (call

him Tom) that she likes less than her spouse in The Mating Ritual (call him Keith). This means that

Nicole prefers Keith to Tom. (+)

By Theorem 5.2.18 and the fact that Nicole and Keith are married in the Mating Ritual, we know that

Keith prefers Nicole to his spouse in $\mathcal{M}$. (++)

This means that Keith and Nicole form a rogue couple in $\mathcal{M}$, which contradicts the stability of $\mathcal{M}$. ■

Applications

Not surprisingly, a stable matching procedure is used by at least one large on-line dating agency. But although "man-woman-marriage" terminology is traditional and makes some of the definitions easier to remember (we hope without offending anyone), solu-

tions to the Stable Marriage Problem are widely useful.

The Mating Ritual was first announced in a paper by D. Gale and L.S. Shapley in 1962, but ten years before the Gale-Shapley paper was appeared, and unknown by them, the Ritual was being used to assign residents to hospitals by the National Resident Matching Program (NRMP). The NRMP has, since the turn of the twentieth century, assigned each year's pool of medical school graduates to hospital residencies (formerly called "internships") with hospitals and graduates playing the roles of men and women. (In this case, there may be multiple women married to one man, a scenario we consider in the problem section at the end of the chapter.). Before the Ritual was adopted, there were chronic disruptions and awkward countermeasures taken to preserve assignments of graduates to residencies. The Ritual resolved these problems so successfully, that it was used essentially without change at least through 1989.[10]

[10]Much more about the Stable Marriage Problem can be found in the very readable mathematical mono-

The internet infrastructure company, Akamai, also uses a variation of the Mating Ritual to assign web traffic to its servers. In the early days, Akamai used other combinatorial optimization algorithms that got to be too slow as the number of servers (over 65,000 in 2010) and requests (over 800 billion per day) increased. Akamai switched to a Ritual-like approach ~~The Mating Ritual~~ since it is fast and can be run in a distributed manner. In this case, web requests correspond to women and web servers correspond to men. The web requests have preferences based on latency and packet loss, and the web servers have preferences based on cost of bandwidth and colocation.

graph by Dan Gusfield and Robert W. Irving, The Stable Marriage Problem: Structure and Algorithms, MIT Press, Cambridge, Massachusetts, 1989, 240 pp.

Not surprisingly, the Mating Ritual is also used by at least one large on-line dating agency. Even here, ~~once again~~, there is no serenading going on — everything is handled by computer.

5.3 Coloring

In Section 5.2, we used edges to indicate an affinity between a pair of nodes. We now consider situations where it is useful to use edges to represent a *conflict* between a pair of nodes. For example, consider the following exam scheduling problem.

5.3.1 An Exam Scheduling Problem

Each term, the MIT Schedules Office must assign a time slot for each final exam. This is not easy, because some students are taking several classes with finals, and (even at MIT) a student can take only one test during a particular time slot. The Schedules Office wants to avoid all conflicts. Of course, you can make such a schedule by having every exam in a different slot, but then you would need hundreds of slots for the hundreds of

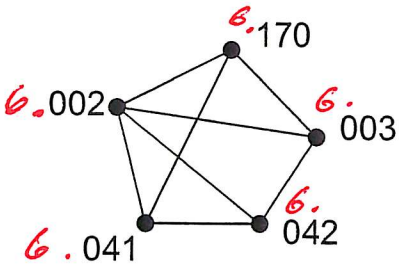


Figure 5.15: A scheduling graph for five exams. Exams connected by an edge cannot be given at the same time.

courses, and the exam period would run all year! So, the Schedules Office would also like to keep exam period short.

The Schedules Office's problem is easy to describe as a graph. There will be a vertex for each course with a final exam, and two vertices will be adjacent exactly when some student is taking both courses. For example, suppose we need to schedule exams for 6.041, 6.042, 6.002, 6.003 and 6.170. The scheduling graph might appear as in Figure 5.15.

6.002 and 6.042 cannot have an exam at the same time since there are students in both courses, so there is an edge between their nodes. On the other hand, 6.042 and 6.170 can have an exam at the same time if they're taught at the same time (which they sometimes are), since no student can be enrolled in both (that is, no student *should* be enrolled in both when they have a timing conflict).

We next identify each time slot with a color. For example, Monday morning is red, Monday afternoon is blue, Tuesday morning is green, etc. Assigning an exam to a time slot is then equivalent to coloring the corresponding vertex. The main constraint is that *adjacent vertices must get different colors*—otherwise, some student has two exams at the same time. Furthermore, in order to keep the exam period short, we should try to color all the vertices using as *few different colors as possible.* As shown in Figure 5.16, three colors suffice for our example.

The coloring in Figure 5.16 corresponds to giving one final on Monday morning (red),

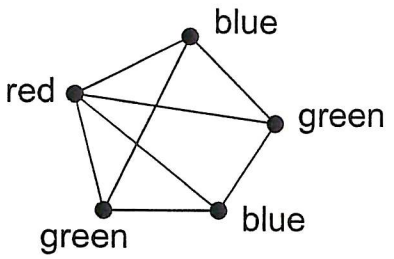


Figure 5.16: A 3-coloring of the exam graph from Figure 5.15.

two Monday afternoon (blue), and two Tuesday morning (green). Can we use fewer than three colors? No! We can't use only two colors since there is a triangle in the graph, and three vertices in a triangle must all have different colors.

This is an example of what is a called a *graph coloring problem*: given a graph G, assign colors to each node such that adjacent nodes have different colors. A color assignment with this property is called a *valid coloring* of the graph—a "*coloring*," for short. A graph G is *k-colorable* if it has a coloring that uses at most k colors.

Definition 5.3.1. The minimum value of k for which a graph, G, has a valid coloring is called its *chromatic number*, $\chi(G)$.

In general, trying to figure out if you can color a graph with a fixed number of colors can take a long time. It's a classic example of a problem for which no fast algorithms are known. In fact, it is easy to check if a coloring works, but it seems really hard to find it. (If you figure out how, then you can get a $1 million Clay prize.)

5.3.2 Degree-Bounded Coloring

There are some simple graph properties that give useful upper bounds on the chromatic number. For example, if we have an upper bound on the degrees of all the vertices in a graph, then we can easily find a coloring with only one more color than the degree bound.

Theorem 5.3.2. *A graph with maximum degree at most k is $(k+1)$-colorable.*

The natural way to try to prove this theorem is to use induction on k. Unfortunately, this approach leads to disaster. It is not that it is impossible, just that it is extremely painful and would ruin your week if you tried it on an exam. When you encounter such a disaster when using induction on graphs, it is usually best to change what you are inducting on. In graphs, some good choices are n, the number of nodes, or e, the number of edges.

Proof of Theorem 5.3.2. We use induction on the number of vertices in the graph, which we denote by n. Let $P(n)$ be the proposition that an n-vertex graph with maximum degree at most k is $(k+1)$-colorable.

Base case ($n = 1$): A 1-vertex graph has maximum degree 0 and is 1-colorable, so $P(1)$ is true.

Inductive step: Now assume that $P(n)$ is true, and let G be an $(n+1)$-vertex graph with

maximum degree at most k. Remove a vertex v (and all edges incident to it), leaving an n-vertex subgraph, H. The maximum degree of H is at most k, and so H is $(k+1)$-colorable by our assumption $P(n)$. Now add back vertex v. We can assign v a color (from the set of $k+1$ colors) that is different from all its adjacent vertices, since there are at most k vertices adjacent to v and so at least one of the $k+1$ colors is still available. Therefore, G is $(k+1)$-colorable. This completes the inductive step, and the theorem follows by induction. ■

Sometimes $k+1$ colors is the best you can do. For example, in the complete graph, K_n, every one of its n vertices is adjacent to all the others, so all n must be assigned different colors. Of course n colors is also enough, so $\chi(K_n) = n$. In this case, every node has degree $k = n-1$ and so this is an example where Theorem 5.3.2 gives the best possible bound. By a similar argument, we can show that Theorem 5.3.2 gives the best

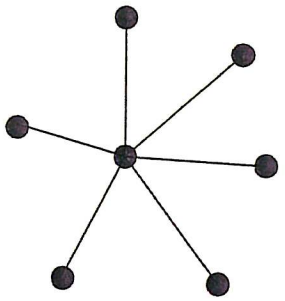


Figure 5.17: A 7-node star graph.

possible bound for *any* graph with degree bounded by k that has K_{k+1} as a subgraph.

But sometimes $k + 1$ colors is far from the best that you can do. For example, the n-node *star graph* shown in Figure 5.17 has maximum degree $n - 1$ but can be colored using just 2 colors.

5.3.3 Why coloring?

One reason coloring problems come all the time is because scheduling conflicts are so common. For example, at Akamai, a new version of software is deployed over each of

65,000 servers every few days. The updates cannot be done at the same time since the servers need to be taken down in order to deploy the software. Also, the servers cannot be handled one at a time, since it would take forever to update them all (each one takes about an hour). Moreover, certain pairs of servers cannot be taken down at the same time since they have common critical functions. This problem was eventually solved by making a 65,000 node conflict graph and coloring it with 8 colors—so only 8 waves of install are needed!

Another example comes from the need to assign frequencies to radio stations. If two stations have an overlap in their broadcast area, they can't be given the same frequency. Frequencies are precious and expensive, so you want to minimize the number handed out. This amounts to finding the minimum coloring for a graph whose vertices are the stations and whose edges connect stations with overlapping areas.

Coloring also comes up in allocating registers for program variables. While a variable

is in use, its value needs to be saved in a register. Registers can be reused for different variables but two variables need different registers if they are referenced during overlapping intervals of program execution. So register allocation is the coloring problem for a graph whose vertices are the variables; vertices are adjacent if their intervals overlap, and the colors are registers. Once again, the goal is to minimize the number of colors needed to color the graph.

Finally, there's the famous map coloring problem stated in Proposition 1.3.4. The question is how many colors are needed to color a map so that adjacent territories get different colors? This is the same as the number of colors needed to color a graph that can be drawn in the plane without edges crossing. A proof that four colors are enough for *planar* graphs was acclaimed when it was discovered about thirty years ago. Implicit in that proof was a 4-coloring procedure that takes time proportional to the number of vertices in the graph (countries in the map). Surprisingly, it's another of those million

dollar prize questions to find an efficient procedure to tell if a planar graph really *needs* four colors or if three will actually do the job. (It's always easy to tell if an *arbitrary* graph is 2-colorable.) Later in this chapter, we'll develop enough planar graph theory to present an easy proof that all planar graphs are 5-colorable.

5.4 Getting from A to B in a Graph

5.4.1 Paths and Walks

Definition 5.4.1. A *walk*[11] in a graph, G, is a sequence of vertices

$$v_0, v_1, \ldots, v_k$$

and edges

$$v_0\text{–}v_1, v_1\text{–}v_2, \ldots, v_{k-1}\text{–}v_k$$

[11] Some texts use the word *path* for our definition of walk and the term *simple path* for our definition of path.

such that v_i–v_{i+1} is an edge of G for all i where $0 \leq i < k$. The walk is said to *start* at v_0 and to *end* at v_k, and the *length* of the walk is defined to be k. An edge, u–v, is *traversed* n times by the walk if there are n different values of i such that v_i–$v_{i+1} = u$–v. A *path* is a walk where all the v_i's are different, that is, $i \neq j$ implies $v_i \neq v_j$. For simplicity, we will refer to paths and walks by the sequence of vertices.[12]

For example, the graph in Figure 5.18 has a length 6 path A, B, C, D, E, F, G. This is the longest path in the graph. Of course, the graph has walks with arbitrarily large lengths (*e.g.*, ABABAB...).

The length of a walk or path is the total number of times it traverses edges, which is *one less* than its length as a sequence of vertices. For example, the length 6 path A, B, C,

[12] This works fine for simple graphs since the edges in a walk are completely determined by the sequence of vertices and there is no ambiguity. For graphs with multiple edges, we would need to specify the edges as well as the nodes.

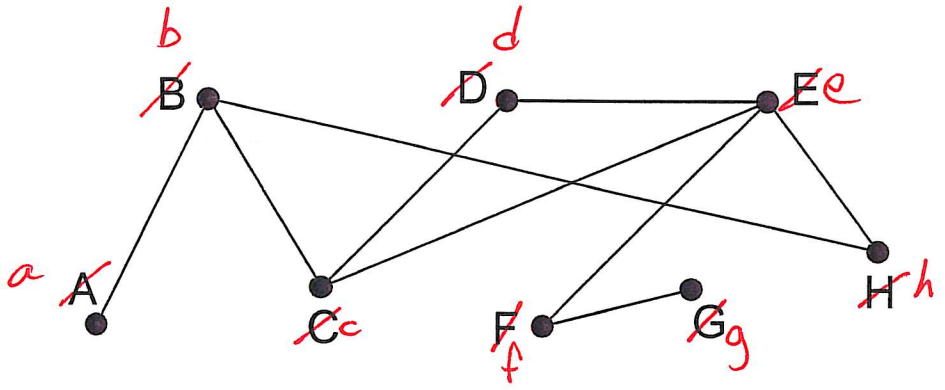


Figure 5.18: A graph containing a path A, B, C, D, E, F, G of length 6.

D, E, F, G contains a sequence of 7 vertices.

5.4.2 Finding a Path

Where there's a walk, there's a path. This is sort of obvious, but it's easy enough to prove rigorously using the Well Ordering Principle.

Lemma 5.4.2. *If there is a walk from a vertex u to a vertex v in a graph, then there is a path from u to v.*

Proof. Since there is a walk from u to v, there must, by the Well-ordering Principle, be a minimum length walk from u to v. If the minimum length is zero or one, this minimum length walk is itself a path from u to v. Otherwise, there is a minimum length walk

$$v_0, v_1, \ldots, v_k$$

from $u = v_0$ to $v = v_k$ where $k \geq 2$. We claim this walk must be a path. To prove the claim, suppose to the contrary that the walk is not a path, that is, some vertex on the walk occurs twice. This means that there are integers i, j such that $0 \leq i < j \leq k$ with $v_i = v_j$. Then deleting the subsequence

$$v_{i+1}, \ldots, v_j$$

yields a strictly shorter walk

$$v_0, v_1, \ldots, v_i, v_{j+1}, v_{j+2}, \ldots, v_k$$

from u to v, contradicting the minimality of the given walk. ∎

Actually, we proved something stronger:

Corollary 5.4.3. *For any walk of length k in a graph, there is a path of length* at most *k with the same endpoints. Moreover, the shortest walk between a pair of vertices is, in fact, a path.*

5.4.3 Numbers of Walks

Given a pair of nodes that are connected by a walk of length k in a graph, there are often many walks that can be used to get from one node to the other. For example, there are 5 walks of length 3 that start at v_1 and end at v_4 in the graph shown in Figure 5.19.

Redraft graphic: Missing graphic

Figure 5.19: A graph for which there are 5 walks of length 3 from v_1 to v_4. The walks are (v_1, v_2, v_1, v_4), (v_1, v_3, v_1, v_4), (v_1, v_4, v_1, v_4), (v_1, v_2, v_3, v_4), and (v_1, v_4, v_3, v_4).

There is a surprising relationship between the number of walks of length k between

a pair of nodes in a graph G and the kth power of the adjacency matrix A_G for G. The relationship is captured in the following theorem.

Theorem 5.4.4. *Let $G = (V, E)$ be an n-node graph with $V = \{v_1, v_2, \ldots, v_n\}$ and let $A_G = \{a_{ij}\}$ denote the adjacency matrix for G. Let $a_{ij}^{(k)}$ denote the (i, j)-entry of the kth power of A_G. Then the number of walks of length k between v_i and v_j is $a_{ij}^{(k)}$.*

In other words, we can determine the number of walks of length k between any pair of nodes simply by computing the kth power of the adjacency matrix! That's pretty amazing.

For example, the first three powers of the adjacency matrix for the graph in Figure 5.19 are:

$$A = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \qquad A^2 = \begin{pmatrix} 3 & 1 & 2 & 1 \\ 1 & 2 & 1 & 2 \\ 2 & 1 & 3 & 1 \\ 1 & 2 & 1 & 2 \end{pmatrix} \qquad A^3 = \begin{pmatrix} 4 & 5 & 5 & 5 \\ 5 & 2 & 5 & 2 \\ 5 & 5 & 4 & 5 \\ 5 & 2 & 5 & 2 \end{pmatrix}$$

Sure enough, the $(1, 4)$ coordinate of A^3 is $a_{14}^{(3)} = 5$, which is the number of length 3

walks from v_1 to v_4. And $a_{24}^{(3)} = 2$, which is the number of length 3 walks from v_2 to v_4. By proving the theorem, we'll discover why it is true and thereby uncover the relationship between matrix multiplication and numbers of walks.

Proof of Theorem 5.4.4. The proof is by induction on k. We will let $P(k)$ be the predicate that the theorem is true for k. Let $P_{ij}^{(k)}$ denote the number of walks of length k between v_i and v_j. Then $P(k)$ is the predicate

$$\forall i, j \in [1, n].\ P_{ij}^{(k)} = a_{ij}^{(k)}. \tag{5.2}$$

Base Case ($k = 1$): There are two cases to consider:

Case 1: $\{v_i, v_j\} \in E$. Then $P_{ij}^{(1)} = 1$ since there is precisely one walk of length 1 between v_i and v_j. Moreover, $\{v_i, v_j\} \in E$ means that $a_{ij}^{(1)} = a_{ij} = 1$. So, $P_{ij}^{(1)} = a_{ij}^{(1)}$ in this case.

Case 2: $\{v_i, v_j\} \notin E$. Then $P_{ij}^{(1)} = 0$ since there cannot be any walks of length 1 between

v_i and v_j. Moreover, $\{v_i, v_j\} \notin E$ means that $a_{ij} = 0$. So, $P_{ij}^{(1)} = a_{ij}^{(1)}$ in this case as well.

Hence, $P(1)$ must be true.

remove space

Inductive Step: Assume $P(k)$ is true. In other words, assume that equation 5.2 holds.

We can group (and thus count the number of) walks of length $k+1$ from v_i to v_j according to the first edge in the walk (call it $\{v_i, v_t\}$). This means that

$$P_{ij}^{(k+1)} = \sum^{t:\{v_i,v_t\}\in E} P_{tj}^{(k)} \tag{5.3}$$

where the sum is over all t such that $\{v_i, v_t\}$ is an edge. Using the fact that $a_{ij} = 1$ if $\{v_i, v_t\} \in E$ and $a_{it} = 0$ otherwise, we can rewrite Equation 5.3 as follows:

$$P_{ij}^{(k+1)} = \sum_{t=1}^{n} a_{it} P_{tj}^{(k)}.$$

By the inductive hypothesis, $P_{ij}^{(k)} = a_{ij}^{(k)}$ and thus

$$P_{ij}^{(k+1)} = \sum_{t=1}^{n} a_{it} a_{tj}^{(k)}.$$

But the formula for matrix multiplication gives that

$$a_{ij}^{(k+1)} = \sum_{t=1}^{n} a_{it} a_{tj}^{(k)}.$$

and so we must have $P_{ij}^{(k+1)} = a_{ij}^{(k+1)}$ for all $i, j \in [1, n]$. Hence $P(k+1)$ is true and the induction is complete. ■

5.4.4 Shortest Paths

Although the connection between the power of the adjacency matrix and the number of walks is cool (at least if you are a mathematician), the problem of counting walks does not come up very often in practice. Much more important is the problem of finding the shortest path between a pair of nodes in a graph.

There is good news and bad news to report on this front. The good news is that it is not very hard to find a shortest path. The bad news is that you can't win one of those million dollar prizes for doing it.

In fact, there are several good algorithms known for finding a Shortest Path between a pair of nodes. The simplest to explain (but not the fastest) is to compute the powers of the adjacency matrix one by one until the value of $a_{ij}^{(k)}$ exceeds 0. That's because Theorem 5.4.4 and Corollary 5.4.3 imply that the length of the shortest path between v_i and v_j will be the smallest value of k for which $a_{ij}^{(k)} > 0$.

Paths in Weighted Graphs

The problem of computing shortest paths in a weighted graph frequently arises in practice. For example, when you drive home for vacation, you usually would like to take the shortest route.

Definition 5.4.5. Given a weighted graph, the length of a path in the graph is the sum of the weights of the edges in the path.

Finding shortest paths in weighted graphs is not a lot harder than finding shortest paths in unweighted graphs. We won't show you how to do it here, but you will study algorithms for finding shortest paths if you take an algorithms course. Not surprisingly, the proof of correctness will use induction.

5.4.5 Connectivity

Definition 5.4.6. Two vertices in a graph are said to be *connected* when there is a path that begins at one and ends at the other. By convention, every vertex is considered to be connected to itself by a path of length zero.

Definition 5.4.7. A graph is said to be *connected* when every pair of vertices are connected.

5.4.6 Connected Components

Being connected is usually a good property for a graph to have. For example, it could mean that it is possible to get from any node to any other node, or that it is possible to communicate between any pair of nodes, depending on the application.

But not all graphs are connected. For example, the graph where nodes represent cities and edges represent highways might be connected for North America cities, but would surely not be connected if you also included cities in Australia. The same is true for communication networks like the Internet—in order to be protected from viruses that spread on the Internet, some government networks are completely isolated from the Internet.

For example, the diagram in Figure 5.20 looks like a picture of three graphs, but is intended to be a picture of *one* graph. This graph consists of three pieces (subgraphs).

Figure 5.20: One graph with 3 connected components.

Each piece by itself is connected, but there are no paths between vertices in different pieces. These connected pieces of a graph are called its *connected components*.

Definition 5.4.8. A *connected component* is a subgraph of a graph consisting of some vertex and every node and edge that is connected to that vertex.

So a graph is connected iff it has exactly one connected component. The empty graph on n vertices has n connected components.

5.5.2

5.4.7 k-Connected Graphs

If we think of a graph as modelling cables in a telephone network, or oil pipelines, or electrical power lines, then we not only want connectivity, but we want connectivity that survives component failure. A graph is called *k-edge connected* if it takes at least k "edge-failures" to disconnect it. More precisely:

Definition 5.4.9. Two vertices in a graph are *k-edge connected* if they remain connected in every subgraph obtained by deleting $k-1$ edges. A graph with at least two vertices is k-edge connected[13] if every two of its vertices are k-edge connected.

So 1-edge connected is the same as connected for both vertices and graphs. Another way to say that a graph is k-edge connected is that every subgraph obtained from it by deleting at most $k-1$ edges is connected. For example, in the graph in Figure 5.18,

[13]The corresponding definition of connectedness based on deleting vertices rather than edges is common in Graph Theory texts and is usually simply called "k-connected" rather than "k-vertex connected."

vertices C and E are 3-edge connected, B and E are 2-edge connected, G and E are 1-edge connected, and no vertices are 4-edge connected. The graph as a whole is only 1-edge connected. The complete graph, K_n, is $(n-1)$-edge connected.

If two vertices are connected by k edge-disjoint paths (that is, no two paths traverse the same edge), then they are obviously k-edge connected. A fundamental fact, whose ingenious proof we omit, is Menger's theorem which confirms that the converse is also true: if two vertices are k-edge connected, then there are k edge-disjoint paths connecting them. It even takes some ingenuity to prove this for the case $k = 2$.

5.4.8 The Minimum Number of Edges in a Connected Graph

The following theorem says that a graph with few edges must have many connected components.

Theorem 5.4.10. *Every graph with v vertices and e edges has at least $v - e$ connected compo-*

nents.

Of course for Theorem 5.4.10 to be of any use, there must be fewer edges than vertices.

Proof. We use induction on the number of edges, e. Let $P(e)$ be the proposition that

> for every v, every graph with v vertices and e edges has at least $v - e$ connected components.

Base case:($e = 0$). In a graph with 0 edges and v vertices, each vertex is itself a connected component, and so there are exactly $v = v - 0$ connected components. So $P(e)$ holds.

Inductive step: Now we assume that the induction hypothesis holds for every e-edge graph in order to prove that it holds for every $(e+1)$-edge graph, where $e \geq 0$. Consider a graph, G, with $e+1$ edges and v vertices. We want to prove that G has at least $v-(e+1)$ connected components. To do this, remove an arbitrary edge a–b and call the resulting

graph G'. By the induction assumption, G' has at least $v - e$ connected components. Now add back the edge a–b to obtain the original graph G. If a and b were in the same connected component of G', then G has the same connected components as G', so G has at least $v - e > v - (e+1)$ components. Otherwise, if a and b were in different connected components of G', then these two components are merged into one component in G, but all other components remain unchanged, reducing the number of components by 1. Therefore, G has at least $(v - e) - 1 = v - (e + 1)$ connected components. So in either case, $P(e + 1)$ holds. This completes the Inductive step. The theorem now follows by induction. ■

Corollary 5.4.11. *Every connected graph with v vertices has at least $v - 1$ edges.*

A couple of points about the proof of Theorem 5.4.10 are worth noticing. First, we used induction on the number of edges in the graph. This is very common in proofs

involving graphs, ~~and so~~ as is induction on the number of vertices. When you're presented with a graph problem, these two approaches should be among the first you consider.

The second point is more subtle. Notice that in the inductive step, we took an arbitrary $(n+1)$-edge graph, threw out an edge so that we could apply the induction assumption, and then put the edge back. You'll see this shrink-down, grow-back process very often in the inductive steps of proofs related to graphs. This might seem like needless effort; why not start with an n-edge graph and add one more to get an $(n+1)$-edge graph? That would work fine in this case, but opens the door to a nasty logical error called *buildup error*.

5.4.9 Build-Up Error

False Claim. *If every vertex in a graph has degree at least 1, then the graph is connected.*

There are many counterexamples; for example, see Figure 5.21.

Redraft graphic: Missing graphic

Figure 5.21: A counterexample graph to the False Claim

False proof. We use induction. Let $P(n)$ be the proposition that if every vertex in an n-vertex graph has degree at least 1, then the graph is connected.

Base case: There is only one graph with a single vertex and has degree 0. Therefore, $P(1)$ is vacuously true, since the if-part is false.

Inductive step: We must show that $P(n)$ implies $P(n+1)$ for all $n \geq 1$. Consider an n-vertex graph in which every vertex has degree at least 1. By the assumption $P(n)$, this graph is connected; that is, there is a path between every pair of vertices. Now we add one more vertex x to obtain an $(n+1)$-vertex graph as shown in Figure 5.22.

Redraft graphic: Missing graphic

Figure 5.22: Adding a vertex x with degree at least 1 to a connected n-node graph.

All that remains is to check that there is a path from x to every other vertex z. Since x has degree at least one, there is an edge from x to some other vertex; call it y. Thus, we can obtain a path from x to z by adjoining the edge x–y to the path from y to z. This proves $P(n+1)$.

By the principle of induction, $P(n)$ is true for all $n \geq 1$, which proves the theorem ■

That looks fine! Where is the bug? It turns out that the faulty assumption underlying this argument is that *every $(n+1)$-vertex graph with minimum degree 1 can be obtained from an n-vertex graph with minimum degree 1 by adding 1 more vertex.* Instead of starting by considering an arbitrary $(n+1)$-node graph, this proof only considered $(n+1)$-node graphs that you can make by starting with an n-node graph with minimum degree 1.

The counterexample above shows that this assumption is false; there is no way to build the 4-vertex graph in Figure 5.21 from a 3-vertex graph with minimum degree 1.

Thus the first error in the proof is the statement "This proves $P(n+1)$."

This kind of flaw is known as "build-up error." Usually, build-up error arises from a faulty assumption that every size $n+1$ graph with some property can be "built up" from a size n graph with the same property. (This assumption is correct for some properties, but incorrect for others—such as the one in the argument above.)

One way to avoid an accidental build-up error is to use a "shrink down, grow back" process in the inductive step: *i.e.*, start with a size $n+1$ graph, remove a vertex (or edge), apply the inductive hypothesis $P(n)$ to the smaller graph, and then add back the vertex (or edge) and argue that $P(n+1)$ holds. Let's see what would have happened if we'd tried to prove the claim above by this method:

Revised inductive step: We must show that $P(n)$ implies $P(n+1)$ for all $n \geq 1$. Consider an $(n+1)$-vertex graph G in which every vertex has degree at least 1. Remove an

arbitrary vertex v, leaving an n-vertex graph G' in which every vertex has degree... uh oh!

The reduced graph G' might contain a vertex of degree 0, making the inductive hypothesis $P(n)$ inapplicable! We are stuck—and properly so, since the claim is false!

Always use shrink-down, grow-back arguments and you'll never fall into this trap.

5.5 Around and Around We Go

5.5.1 Cycles and Closed Walks

Definition 5.5.1. A *closed walk*[14] in a graph G is a sequence of vertices

$$v_0, v_1, \ldots, v_k$$

[14]Some texts use the word *cycle* for our definition of closed walk and *simple cycle* for our definition of cycle.

and edges

$$v_0\text{–}v_1, v_1\text{–}v_2, \ldots, v_{k-1}\text{–}v_k$$

where v_0 is the same node as v_k and v_i–v_{i+1} is an edge of G for all i where $0 \leq i < k$.

The *length* of the closed walk is k. A closed walk is said to be a *cycle* if $k \geq 3$ and $v_0, v_1, \ldots, v_{k-1}$ are all different.

For example, B, C, D, E, C, B is a closed walk of length 5 in the graph shown in Figure 5.18. It is not a cycle since it contains node C twice. On the other hand, C, D, E, C is a cycle of length 3 in this graph since every node appears just once.

There are many ways to represent the same closed walk or cycle. For example, B, C, D, E, C, B is the same as C, D, E, C, B, C (just starting at node C instead of node B) and the same as B, C, E, D, C, B (just reversing the direction).

Cycles are similar to paths, except that the last node is the first node and the notion of

first and last does not matter. Indeed, there are many possible vertex orders that can be used to describe cycles and closed walks, whereas walks and paths have a prescribed beginning, end, and ordering.

INSERT X & Y geoshed

5.5.2 Euler Tours

Can you walk every hallway in the Museum of Fine Arts *exactly once*? If we represent hallways and intersections with edges and vertices, then this reduces to a question about graphs. For example, could you visit every hallway exactly once in a museum with the floor plan in Figure 5.23?

The entire field of graph theory began when Euler asked whether the seven bridges of Königsberg could all be traversed exactly once—essentially the same question we asked about the Museum of Fine Arts. In his honor, an *Euler walk* is a defined to be a walk that traverses every edge in a graph exactly once. Similarly, an *Euler tour* is an Euler walk

5.6.2 Odd cycles and 2-Colorability

We have already seen that determining the chromatic number of a graph is a challenging problem. There is one special case where this problem is very easy; namely the case where every cycle in the graph has even length. In this case, the graph is 2-colorable! Of course, this is optimal if the graph has any edges at all. More generally,

we will prove:

Theorem XY: The following properties of a graph are equivalent (i.e., if the graph has any one of the properties, then it has all of the properties):

i) the graph is bipartite,
ii) the graph is 2-colorable,
iii) the graph does not contain any closed walks with odd length

(iv) the graph does not contain any cycles with odd length.

Proof: we will show that property i IMPLIES property ii, ii IMPLIES iii, iii IMPLIES iv, and iv IMPLIES i. This will show that all four properties are equivalent by repeated application of Rule ____ in Section 2.__ .

i IMPLIES ii: Assume that $G = (V, E)$ is a bipartite graph. Then V can be partitioned into two sets L and R so that no edge connects a pair of nodes in L nor a pair of nodes in R. Hence, we can use one color for all the nodes in L and a second color for all the nodes in R. Hence $\chi(G) = 2$.

ii IMPLIES iii: Let $G = (V, E)$ be a 2-colorable graph and

$W := v_0, v_1, \ldots, v_k$

Be any closed walk in G. Consider any ~~odd~~ 2-coloring for the nodes of G. Since $v_i - v_{i+1}$ ~~is an edge of G~~ $\in E$, v_i and v_{i+1} must be differently colored, for $0 \leq i < k$.

Hence, $v_0, v_2, v_4, \ldots$ have one color and $v_1, v_3, v_5, \ldots$ have the other color. Since W is a closed walk, ~~$v_k = v_0$~~ v_k is the same node as v_0 and so ~~k~~ k must be an even number. This means that W has even length.

iii IMPLIES iv: Since every cycle is a closed walk, ~~this implication~~ a graph without any odd-length closed walks cannot have any odd-length cycles.

iv IMPLIES i: This is the hardest implication to prove. Let G = (V, E) be a graph that does not contain any odd cycles. We will show that every connected component of G is bipartite, which will mean that G is ~~not~~ bipartite.

Let G' = (V', E') be any connected component of G and

let v be ~~some~~ some node in V'. For ~~every~~ every other node $u \in V'$, define

$dist(u)$ ::= the length of the shortest path from u to v in G'. If $u = v$, the distance is zero.

Partition V' into sets L and R so that

$L = \{ u \mid dist(u) \text{ is even} \}$,

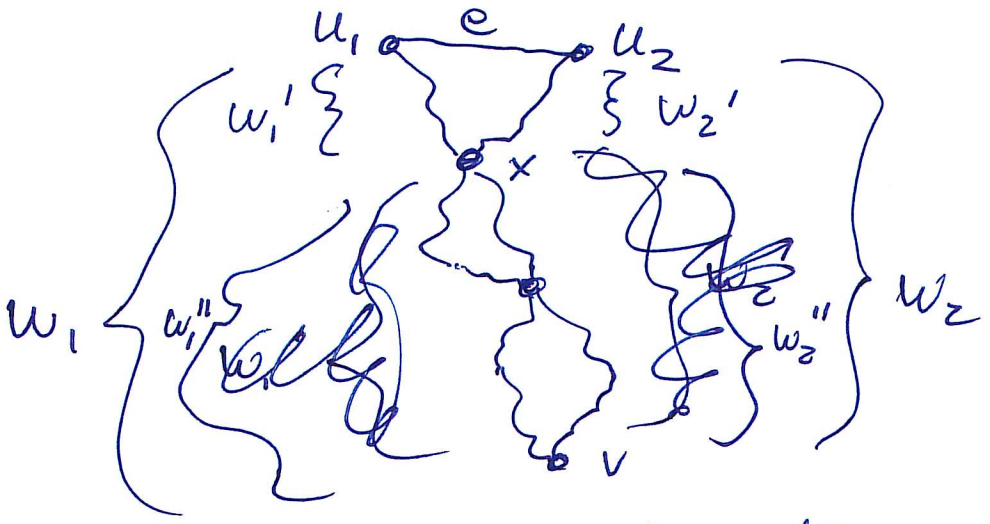
$R = \{ u \mid dist(u) \text{ is odd} \}$.

We will show that G' is bipartite by showing that no edge connects a pair of nodes in L nor a pair of nodes in R.

The proof is by contradiction. Assume their is an edge e that connects a pair of nodes u_1 and u_2 that are both in L or both in R. ~~(The argument is the same if u_1 and u_2 are both in R.)~~ Let W_1 be a shortest path from u_1 to v and let W_2 be a shortest path from u_2 to v. Let x be the first node on the path from u_1 to v that is also on W_2.

Since u_1 and u_2 are both in L or both in R, this means that the lengths of W_1 and W_2 have the same parity (even or odd)

Such a node must exist by the wellordering principle since v is in both W_1 and W_2. ~~These definitions are illustrated in~~ Let w_i' be the subpath of W_i from u_i to x and w_i'' be the subpath of W_i from x to v. for $i=1,2$. These definitions are illustrated in Figure XY3.



For $i=1$ and 2,

Figure XY3 : W_i is the shortest path from u_i to v. x is the first node on u_1 that is contained in W_2. W_i' is the subpath of W_i from u_i to x, and W_i'' is the subpath of W_i from x to v.

Since ~~a~~ w_1 and w_2 are shortest paths, it must be that w_1'' and w_2'' have equal lengths, ~~otherwise we could find a shorter pa~~ ~~short~~ ~~then~~

~~For example, if w_1' is~~ ~~If w_1'' had shorter length,~~ For example, if w_1'' was ~~were~~ shorter than w_2'', then we could combine ~~the~~ w_2' ~~path from x~~ with ~~end~~ ~~the~~ w_1'' to produce a shorter walk ~~path~~ from v_2 to v than w_2, which is not possible.

~~Similarly w_1' and w_2' must have the same length.~~

~~In addition, the length of~~

Similarly, w_1' and w_2' must have the same length. The proof is by contradiction. Suppose they have different length. There are then ~~two~~ two cases.

Case 1: The lengths of w_1' and w_2' differ by 1. Since the lengths of w_1'' and w_2'' are equal, this means that the lengths of w_1 and w_2 differ by 1. This is a contradiction since the ~~pa~~ lengths of w_1 and w_2 have the same parity.

Case 2: The lengths of w_1' and w_2' differ by at least 2. For example, suppose w_1' has at least two more edges

~~Since w_1 and w_2 are shortest paths, the~~

than $w_2' \cdot a$. (The argument is the same if w_2' is the longer path.) Then we can construct a shorter path from u_1 to v than w_1 by first traversing e, then w_2', then w_1''. This is a contradiction since w_1 is a shortest path between u_1 and v.

Since both cases yield a contradiction, we know that w_1' and w_2' have the same length. Since w_1' and w_2' are paths whose only common vertex is x, we can therefore form ~~an odd~~ a cycle by combining w_1', w_2' and e. Since w_1' and w_2' have the same length, this cycle must have odd length, which is a contradiction. Thus ~~there is~~ no edge can connect a pair of nodes in L or a pair of nodes in R. Hence, G must be bipartite, as desired.

This completes the proof that property ii IMPLIES property i, and hence the proof of the theorem. ∎

~~Theorem XY turns out to be more useful~~

Theorem XY turns out to be ~~fairly~~ useful since bipartite graphs come up fairly often in practice. We'll see examples when we talk about planar graphs in Section 5.8 and when we talk about packet routing in communication networks in Chapter 6.

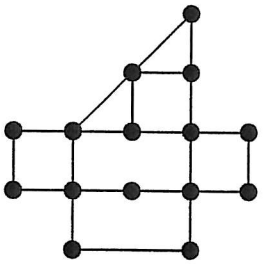


Figure 5.23: A possible floor plan for a museum. Can you find a walk that traverses every edge exactly once?

that starts and finishes at the same vertex. Graphs with Euler tours and Euler walks both have simple characterizations.

Theorem 5.5.2. *A connected graph has an Euler tour if and only if every vertex has even degree.*

Proof. We first show that if a graph has an Euler tour, then every vertex has even degree. Assume that a graph $G = (V, E)$ has an Euler tour $v_0, v_1, \ldots, v_k$ where $v_k = v_0$. Since every edge is traversed once in the tour, $k = |E|$ and the degree of a node u in G is

the number of times that node appears in the sequence $v_0, v_1, \ldots, v_{k-1}$ times two. We multiply by two since if $u = v_i$ for some i where $0 < i < k$, then both v_{i-1}–v_i and v_i–v_{i+1} are edges incident to u in G. If $u = v_0 = v_k$, then both v_{k-1}–v_k and v_0–v_1 are edges incident to u in G. Hence, the degree of every node is even.

We next show that if the degree of every node is even in a graph $G = (V, E)$, then there is an Euler tour. Let

$$W ::= v_0, v_1, \ldots, v_k$$

be the longest walk[15] in G that traverses *no edge more than once*. W must traverse every edge incident to v_k; otherwise the walk could be extended and W would not be the longest walk that traverses all edges at most once. Moreover, it must be that $v_k = v_0$

[15]Did you notice that we are using a variation of the Well Ordering Principle here when we implicitly assume that a longest walk exists? This is ok since the length of a walk where no edge is used more than once is at most $|E|$.

and that W is a closed walk, since otherwise v_k would have odd degree in W (and hence in G), which is not possible by assumption.

We conclude the argument with a proof by contradiction. Suppose that W is not an Euler tour. Because G is a connected graph, we can find an edge not in W but incident to some vertex in W. Call this edge u–v_i. But then we can construct a walk W' that is longer than W but that still uses no edge more than once:

$$W' ::= u, v_i, v_{i+1}, \ldots, v_k, v_1, v_2, \ldots, v_i$$

This contradicts the definition of W, so W must be an Euler tour after all. ■

It is not difficult to extend Theorem 5.5.2 to prove that a connected graph G has an Euler walk if and only if precisely 0 or 2 nodes in G have odd degree. Hence, we can conclude that the graph shown in Figure 5.23 has an Euler walk but not an Euler tour since the graph has precisely two nodes with odd degree.

Although the proof of Theorem 5.5.2 does not explicitly define a method for finding an Euler tour when one exists, it is not hard to modify the proof to produce such a method. The idea is to grow a tour by continually splicing in closed walks until all the edges are consumed.

5.5.3 Hamiltonian Cycles

Hamiltonian cycles are the unruly cousins of Euler tours.

Definition 5.5.3. A *Hamiltonian cycle* in a graph G is a cycle that visits every *node* in G exactly once. Similarly, a *Hamiltonian* path is a path in G that visits every node exactly once.

Although Hamiltonian cycles sound similar to Euler tours—one visits every node once while the other visits every edge once—finding a Hamiltonian cycle can be a lot harder than finding an Euler tour. The same is true for Hamiltonian paths. This is be-

cause no one has discovered a simple characterization of all graphs with a Hamiltonian cycle. In fact, determining whether a graph has a Hamiltonian cycle is the same category of problem as the SAT problem of Section 1.5: you get a million dollars for finding an efficient way to determine when a graph has a Hamiltonian cycle—or proving that no procedure works efficiently on all graphs.

5.5.4 The Travelling Salesperson Problem

As if the problem of finding a Hamiltonian cycle is not hard enough, when the graph is weighted, we often want to find a Hamiltonian cycle that has least possible weight. This is a very famous optimization problem known as the Travelling Salesperson Problem.

Definition 5.5.4. Given a weighted graph G, the *weight* of a cycle in G is defined as the sum of the weights of the edges in the cycle.

For example, consider the graph shown in Figure 5.24 and suppose that you would

Redraft graphic: Missing graphic

Figure 5.24: A weighted graph. Can you find a cycle that visits every node exactly once with weight 15?

like to visit every node once and finish at the node where you started. Can you find way to do this by traversing a cycle with weight 15?

Needless to say, if you can figure out a fast procedure that finds the optimal cycle for the travelling salesperson, you can win a million dollars.

5.6 Trees

As we have just seen, finding good cycles in a graph can be trickier than you might first think. But what if a graph has no cycles at all? Sounds pretty dull. But graphs without cycles (called *acyclic graphs*) are probably the most important graphs of all when it comes

to computer science.

5.6.1 Definitions

Definition 5.6.1. A connected acyclic graph is called a *tree*.

For example, Figure 5.25 shows an example of a 9-node tree.

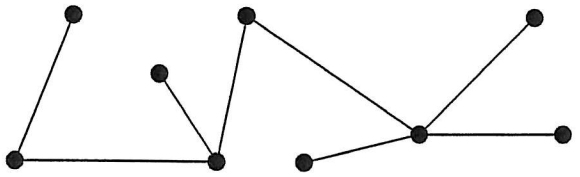


Figure 5.25: A 9-node tree.

The graph shown in Figure 5.26 is not a tree since it is not connected, but it is a forest.

That's because, of course, it consists of a collection of trees.

Redraft graphic: Missing graphic

Figure 5.26: A 6-node forest consisting of 2 component trees. Note that this 6-node graph is not itself a tree since it is not connected.

Definition 5.6.2. If every connected component of a graph G is a tree, then G is a *forest*.

One of the first things you will notice about trees is that they tend to have a lot of nodes with degree one. Such nodes are called *leaves*.

Definition 5.6.3. A *leaf* is a node with degree 1 in a tree (or forest).

For example, the tree in Figure 5.25 has 5 leaves and the forest in Figure 5.26 has 4 leaves.

Trees are a fundamental data structure in computer science. For example, information is often stored in tree-like data structures and the execution of many recursive programs can be modelled as the traversal of a tree. In such cases, it is often useful to draw the

Redraft graphic: Missing graphic

Figure 5.27: The tree from Figure 5.25 redrawn in a leveled fashion, with node E as the root.

tree in a levelled fashion where the node in the top level is identified as the *root*, and where every edge joins a *parent* to a *child*. For example, we have redrawn the tree from Figure 5.25 in this fashion in Figure 5.27. In this example, node D is a child of node E and a parent of nodes B and C.

In the special case of *ordered binary trees*, every node is the parent of at most 2 children and the children are labeled as being a left-child or a right-child.

5.6.2 Properties

Trees have many unique properties. We have listed some of them in the following theorem.

Theorem 5.6.4. *Every tree has the following properties:*

1. *Any connected subgraph is a tree.*
2. *There is a unique simple path between every pair of vertices.*
3. *Adding an edge between nonadjacent nodes in a tree creates a graph with a cycle.*
4. *Removing any edge disconnects the graph.*
5. *If the tree has at least two vertices, then it has at least two leaves.*
6. *The number of vertices in a tree is one larger than the number of edges.*

Proof. 1. A cycle in a subgraph is also a cycle in the whole graph, so any subgraph

of an acyclic graph must also be acyclic. If the subgraph is also connected, then by definition, it is a tree.

2. Since a tree is connected, there is at least one path (see 5.4.7) between every pair of vertices. Suppose for the purposes of contradiction, that there are two different paths between some pair of vertices u and v. Beginning at u, let x be the first vertex where the paths diverge, and let y be the next vertex they share. (For example, see Figure 5.28.) Then there are two paths from x to y with no common edges, which defines a cycle. This is a contradiction, since trees are acyclic. Therefore, there is exactly one path between every pair of vertices.

3. An additional edge u–v together with the unique path between u and v forms a cycle.

4. Suppose that we remove edge u–v. Since the tree contained a unique path between

Figure 5.28: If there are two paths between u and v, the graph must contain a cycle.

u and v, that path must have been u–v. Therefore, when that edge is removed, no path remains, and so the graph is not connected.

5. Let $v_1, \ldots, v_m$ be the sequence of vertices on a longest path in the tree. Then $m \geq 2$, since a tree with two vertices must contain at least one edge. There cannot be an edge v_1–v_i for $2 < i \leq m$; otherwise, vertices $v_1, \ldots, v_i$ would from a cycle. Furthermore, there cannot be an edge u–v_1 where u is not on the path; otherwise, we could make the path longer. Therefore, the only edge incident to v_1 is v_1–v_2, which means that v_1 is a leaf. By a symmetric argument, v_m is a second leaf.

6. We use induction on the number of vertices. For a tree with a single vertex, the claim holds since it has no edges and $0 + 1 = 1$ vertex. Now suppose that the claim holds for all n-vertex trees and consider an $(n+1)$-vertex tree, T. Let v be a leaf of the tree. You can verify that deleting a vertex of degree 1 (and its incident edge) from any connected graph leaves a connected subgraph. So by 1, deleting v and its incident edge gives a smaller tree, and this smaller tree has one more vertex than edge by induction. If we re-attach the vertex, v, and its incident edge, then the equation still holds because the number of vertices and number of edges both increase by 1. Thus, the claim holds for T and, by induction, for all trees. ■

Various subsets of these properties provide alternative characterizations of trees, though we won't prove this. For example, a *connected* graph with a number of vertices one larger than the number of edges is necessarily a tree. Also, a graph with unique paths

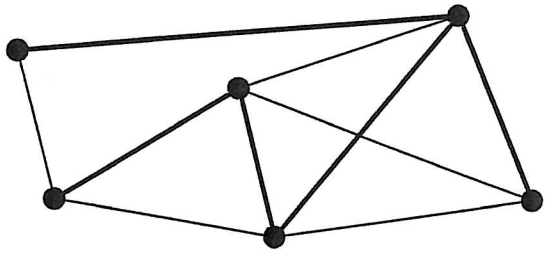


Figure 5.29: A graph where the edges of a spanning tree have been thickened.

between every pair of vertices is necessarily a tree.

5.6.3 Spanning Trees

Trees are everywhere. In fact, every connected graph contains a subgraph that is a tree with the same vertices as the graph. This is a called a *spanning tree* for the graph. For example, Figure 5.29 is a connected graph with a spanning tree highlighted.

Theorem 5.6.5. *Every connected graph contains a spanning tree.*

Proof. By contradiction. Assume there is some connected graph G that has no spanning tree and let T be a connected subgraph of G, with the same vertices as G, and with the smallest number of edges possible for such a subgraph. By the assumption, T is not a spanning tree and so it contains some cycle:

$$v_0\text{–}v_1, v_1\text{–}v_2, \ldots, v_k\text{–}v_0$$

Suppose that we remove the last edge, $v_k\text{–}v_0$. If a pair of vertices x and y was joined by a path not containing $v_k\text{–}v_0$, then they remain joined by that path. On the other hand, if x and y were joined by a path containing $v_k\text{–}v_0$, then they remain joined by a path containing the remainder of the cycle. So all the vertices of G are still connected after we remove an edge from T. This is a contradiction, since T was defined to be a minimum size connected subgraph with all the vertices of G. So the theorem must be true. ■

5.6.4 Minimum Weight Spanning Trees

Spanning trees are interesting because they connect all the nodes of a graph using the smallest possible number of edges. For example the spanning tree for the 6-node graph shown in Figure 5.28 has 5 edges.

Spanning trees are very useful in practice, but in the real world, not all spanning trees are equally desirable. That's because, in practice, there are often costs associated with the edge of the graph.

For example, suppose the nodes of a graph represent buildings or towns and edges represent connections between buildings or towns. The cost to actually make a connection may vary a lot from one pair of buildings or towns to another. The cost might depend on distance or topography. For example, the cost to connect LA to NY might be much higher than that to connect NY to Boston. Or the cost of a pipe through Manhattan

Redraft graphic: Missing graphic

(a)

Redraft graphic: Missing graphic

(b)

Figure 5.30: A spanning tree (a) with weight 19 for graph (b).

might be more than the cost of a pipe through a cornfield.

In any case, we typically represent the cost to connect pairs of nodes with a weighted edge, where the weight of the edge is its cost. The weight of a spanning tree is then just the sum of the weights of the edges in the tree. For example, the weight of the spanning tree shown in Figure 5.30 is 19.

The goal, of course, is to find the spanning tree with minimum weight, called the min-weight tree (MST for short).

Definition 5.6.6. The *min-weight spanning tree* (MST) of an edge-weighted graph G is the spanning tree of G with the smallest possible sum of edge weights.

Redraft graphic: Missing graphic

Figure 5.31: An MST for the graph in Figure 5.30(b) with weight 17.

Is the spanning tree show in Figure 5.30(a) an MST of the weighted graph shown in Figure 5.30(b)? Actually, it is not, since the tree shown in Figure 5.31 is also a spanning tree of the graph shown in Figure 5.30(b), and this spanning tree has weight 17.

What about the graph shown in Figure 5.31? Is it an MST? It seems to be, but how do we prove it? In general, how do we find an MST? We could, of course, enumerate all trees, but this could take forever for very large graphs.

Here are two possible algorithms:

Algorithm 1. *Grow a tree one edge at a time by adding the minimum weight edge of the graph to the tree, making sure that you have a tree at each step.*

Algorithm 2. *1. Select edges one at a time, always choosing the minimum weight edge that*

does not create a cycle with previously-selected edges. We do not impose the constraint that the graph is a tree—our selected edges may indeed form a disconnected graph.

2. *Continue until we get $n - 1$ edges, i.e., a spanning tree.*

For example, in the weighted graph we have been considering, we might run Algorithm 1 as follows. We would start by choosing one of the weight 1 edges, since this is the smallest weight in the graph. Suppose we chose the weight 1 edge on the bottom of the triangle of weight 1 edges in our graph. This edge is incident to two weight 1 edges, a weight 4 edge, a weight 7 edge, and a weight 3 edge. We would then choose the incident edge of minimum weight. In this case, one of the two weight 1 edges. At this point, we cannot choose the third weight 1 edge since this would form a cycle, but we can continue by choosing a weight 2 edge. We might end up with the spanning tree shown in Figure 5.32, which has weight 17, the smallest we've seen so far.

Redraft graphic: Missing graphic

Figure 5.32: A spanning tree found by Algorithm 1.

Now suppose we instead ran Algorithm 2 on our graph. We might again choose the weight 1 edge on the bottom of the triangle of weight 1 edges in our graph. Now, instead of choosing one of the weight 1 edges it touches, we might choose the weight 1 edge on the top of the graph. Note that this edge still has minimum weight, and does not cause us to form a cycle, so Algorithm 2 can choose it. We would then choose one of the remaining weight 1 edges. Note that neither causes us to form a cycle. Continuing the algorithm, we may end up with the same spanning tree in Figure 5.32, though this need not always be the case.

It turns out that both algorithms work, but they might end up with different MSTs. The MST is not necessarily unique—indeed, if all edges of an n-node graph have the

same weight ($= 1$), then all spanning trees have weight $n - 1$.

These are examples of greedy approaches to optimization. Sometimes it works and sometimes it doesn't. The good news is that it works to find the MST. In fact, both variations work. It's a little easier to prove it for Algorithm 2, so we'll do that one here.

Theorem 5.6.7. *For any connected, weighted graph G, Algorithm 2 produces an MST for G.*

Proof. The proof is a bit tricky. We need to show the algorithm terminates, *i.e.*, that if we have selected fewer than $n - 1$ edges, then we can always find an edge to add that does not create a cycle. We also need to show the algorithm creates a tree of minimum weight.

The key to doing all of this is to show that the algorithm never gets stuck or goes in a bad direction by adding an edge that will keep us from ultimately producing an MST. The natural way to prove this is to show that the set of edges selected at any point is

contained in some MST—*i.e.*, we can always get to where we need to be. We'll state this as a Lemma.

Lemma 5.6.8. *For any $m \geq 0$, let S consist of the first m edges selected by Algorithm 2. Then there exists some MST $T = (V, E)$ for G such that $S \subseteq E$,* i.e., *the set of edges that we are growing is always contained in some MST.*

We'll prove this momentarily, but first let's see why it helps prove the theorem. Assume the lemma is true. Then how do we know Algorithm 2 can always find an edge to add without creating a cycle? Well, as long as there are fewer than $n - 1$ edges picked, there exists some edge in $E - S$ and so there is an edge that we can add to S without forming a cycle. Next, how do we know that we get an MST at the end? Well, once $m = n - 1$, we know that S is an MST.

Ok, so the theorem is an easy corollary of the lemma. To prove the lemma, we'll use

induction on the number of edges chosen by the algorithm so far. This is very typical in proving that an algorithm preserves some kind of invariant condition—induct on the number of steps taken, *i.e.*, the number of edges added.

Our inductive hypothesis $P(m)$ is the following: for any G and any set S of m edges initially selected by Algorithm 2, there exists an MST $T = (V, E)$ of G such that $S \subseteq E$.

For the base case, we need to show $P(0)$. In this case, $S = \emptyset$, so $S \subseteq E$ trivially holds for any MST $T = (V, E)$.

For the inductive step, we assume $P(m)$ holds and we show that it implies $P(m+1)$. Let e denote the $(m+1)$st edge selected by Algorithm 2, and let S denote the first m edges selected by Algorithm 2. Let $T^* = (V^*, E^*)$ be the MST such that $S \subseteq E^*$, which exists by the inductive hypothesis. There are now two cases:

Case 1: $e \in E^*$, in which case $S \cup \{e\} \subseteq E^*$, and thus $P(m+1)$ holds.

Redraft graphic: Missing graphic

Figure 5.33: The graph formed by adding e to T^*. Edges of S are denoted with solid lines and edges of $E^* - S$ are denoted with dashed lines.

Case 2: $e \notin E^*$, as illustrated in Figure 5.33. Now we need to find a different MST that contains S and e.

What happens when we add e to T^*? Since T^* is a tree, we get a cycle. (Here we used part 3 of Theorem 5.6.4.) Moreover, the cycle cannot only contains edges in S, since e was chosen so that together with the edges in S, it does not form a cycle. This implies that $\{e\} \cup T^*$ contains a cycle that contains an edge e' of $E^* - S$. For example, such an e' is shown in Figure 5.33.

Note that the weight of e is at most that of e'. Indeed, Algorithm 2 picks the minimum weight edge that does not make a cycle with S. Since $e' \in T^*$, it cannot make a cycle

with S_n and so if the weight of e were greater than the weight of e', Algorithm 2 would not have selected e ahead of e'.

Okay, we're almost done. Now we'll make an MST that contains $S \cup \{e\}$. Let $T^{**} = (V, E^{**})$ where $E^{**} = (E^* - \{e'\}) \cup \{e\}$, that is, we swap e and e' in T^*.

Claim 5.6.9. *T^{**} is an MST.*

Proof of claim. We first show that T^{**} is a spanning tree. T^{**} is acyclic because it was produced by removing an edge from the only cycle in $T^* \cup \{e\}$. T^{**} is connected since the edge we deleted from $T^* \cup \{e\}$ was on a cycle. It follows by Theorem 2? that T^{**} is a spanning tree for G.

Since T^{**} contains all the nodes of G, it must be that

Now let's look at the weight of T^{**}. Well, since the weight of e was at most that of e', the weight of T^{**} is at most that of T^*, and thus T^{**} is an MST for G. ■

Since $S \cup \{e\} \subseteq E^{**}$, $P(m+1)$ holds. Thus, when Algorithm 2 has $n-1$ edges, it must eventually produces an MST. This will happen when it adds $n-1$ edges the the subgraph it builds. ■

So now we know for sure that the MST for our example graph has weight 17 since it was produced by Algorithm 2. And we have a fast algorithm for finding a minimum-weight spanning tree for any graph.

5.7 Planar Graphs

David – pls type in figure captions so I can check them

5.7.1 Drawing Graphs in the Plane

Suppose there are three dogs and three houses, as shown in Figure 5.47. Can you find a route from each dog to each house such that no route crosses any other route?

A *quadapus* is a little-known animal similar to an octopus, but with four arms. Suppose there are five quadapi resting on the sea floor, as shown in Figure 5.35. Can each quadapus simultaneously shake hands with every other in such a way that no arms cross?

David: this is revised

Figure 5.34: Three dog houses and three human houses. Is there a route from each dog house to each human house so that no pair of routes cross each other?

Definition 5.7.1. A *planar graph* is a graph that has a planar drawing. A *drawing of a graph in the plane* consists of an assignment of vertices to distinct points in the plane and an assignment of edges to smooth, non-self-intersecting curves in the plane (whose endpoints are the nodes incident to the edge). The drawing is *planar* (*i.e.*, it is a *planar drawing*) if none of the curves "cross"—*i.e.*, the only points that appear on more than one curve are the vertex points.

Thus, these two puzzles are asking whether the graphs in Figure 5.36 are planar; that is, whether they can be redrawn so that no edges cross. The first graph is called the *complete bipartite graph*, $K_{3,3}$, and the second is K_5.

In each case, the answer is, "No—but almost!" In fact, if you remove an edge from either of them, then the resulting graphs *can* be redrawn in the plane so that no edges

cross. For example, we have illustrated the planar drawings for each resulting graph in Figure 5.37.

Planar drawings have applications in circuit layout and are helpful in displaying graphical data such as program flow charts, organizational charts, and scheduling conflicts. For these applications, the goal is to draw the graph in the plane with as few edge crossings as possible. (See the box on the following page for one such example.)

David: the box is missing

5.7.2 A Recursive Definition for Planar Graphs

Definition 5.7.1 is perfectly precise but has the challenge that it requires us to work with concepts such as a "smooth curve" when trying to prove results about planar graphs. The trouble is that we have not really laid the groundwork from geometry and topology to be able to reason carefully about such concepts. For example, we haven't really defined what it means for a curve to be smooth—we just drew a simple picture (*e.g.*,

Figure 5.37) and hoped you would get the idea.

Relying on pictures to convey new concepts is generally not a good idea and can sometimes lead to disaster (or, at least, false proofs). Indeed, it is because of this issue that there have been so many false proofs relating to planar graphs over time.[17] Such proofs usually rely way too heavily on pictures and have way too many statements like,

> As you can see from Figure ABC, it must be that property XYZ holds for all planar graphs.

The good news is that there is another way to define planar graphs that uses only discrete mathematics. In particular, we can define planar graphs as a recursive data type. In order to understand how it works, we first need to understand the concept of a *face* in a planar drawing.

[17] ~~XYZ's~~ false proof of the 4-Color Theorem for planar graphs is not the only example.

Faces

In a planar drawing of a graph. the curves corresponding to the edges divide up the plane into connected regions. These regions are called the *continuous faces*[18] of the drawing. For example, the drawing in Figure 5.38 has four continuous faces. Face IV, which extends off to infinity in all directions, is called the *outside face*.

Notice that the vertices along the boundary of each of the faces in Figure 5.38 form a cycle. For example, labeling the vertices as in Figure 5.39, the cycles for the face boundaries are

```
abca    abda    bcdb    acda.        (5.4)
```

These four cycles correspond nicely to the four continuous faces in Figure 5.39. So nicely, in fact, that we can identify each of the faces in Figure 5.39 by its cycle. For example,

[18] Most texts drop the word *continuous* from the definition of a face. We need it to differentiate the connected region in the plane from the closed walk in the graph that bounds the region, which [illegible] we will call a discrete face.

the cycle `abca` identifies face III. Hence, we say that the cycles in Equation 5.4 are the *discrete faces* of the graph in Figure 5.39. We use the term "discrete" since cycles in a graph are a discrete data type (as opposed to a region in the plane, which is a continuous data type).

Unfortunately, continuous faces in planar drawings are not always bounded by cycles in the graph—things can get a little more complicated. For example, consider the planar drawing in Figure 5.40. This graph has what we will call a *bridge* (namely, the edge c–e) and the outer face is

`abcefgecda`.

This is not a cycle, since it has to traverse the bridge c–e twice, but it is a closed walk.

As another example, consider the planar drawing in Figure 5.41. This graph has what we will call a *dongle* (namely, the nodes v, x, y, and w, and the edges incident to them)

and the inner face is

```
rstvxyxvwvtur
```

This is not a cycle because it has to traverse *every* edge of the dongle twice—once "coming" and once "going," but once again, it is a closed walk.

It turns out that bridges and dongles are the only complications, at least for connected graphs. In particular, every continuous face in a planar drawing corresponds to a closed walk in the graph. We refer to such closed walks as the *discrete faces* of the drawing.

A Recursive Definition for Planar Embeddings

The association between the continuous faces of a planar drawing and closed walks will allow us to characterize a planar drawing in terms of the closed walks that bound the continuous faces. In particular, it leads us to the discrete data type of *planar embeddings* that we can use in place of continuous planar drawings. Namely, we'll define a planar

embedding recursively to be the set of boundary-tracing closed walks we could get by drawing one edge after another.

Definition 5.7.2. A *planar embedding* of a *connected* graph consists of a nonempty set of closed walks of the graph called the *discrete faces* of the embedding. Planar embeddings are defined recursively as follows:

Base case: If G is a graph consisting of a single vertex, v, then a planar embedding of G has one discrete face, namely the length zero closed walk, v.

Constructor Case (split a face): Suppose G is a connected graph with a planar embedding, and suppose a and b are distinct, nonadjacent vertices of G that appear on some discrete face, γ, of the planar embedding. That is, γ is a closed walk of the form

$$a \ldots b \cdots a.$$

Then the graph obtained by adding the edge a–b to the edges of G has a planar embed-

ding with the same discrete faces as G, except that face γ is replaced by the two discrete faces[19]

$$a \ldots ba \quad \text{and} \quad ab \cdots a,$$

as illustrated in Figure 5.42.

Constructor Case (add a bridge): Suppose G and H are connected graphs with planar embeddings and disjoint sets of vertices. Let a be a vertex on a discrete face, γ, in the embedding of G. That is, γ is of the form

$$a \ldots a.$$

[19] There is a special case of this rule. If G is a line graph beginning with a and ending with b, then the cycles into which γ splits are actually the same. That's because adding edge a–b creates a simple cycle graph, C_n, that divides the plane into an "inner" and an "outer" region with the same border. In order to maintain the correspondence between continuous faces and discrete faces, we have to allow two "copies" of this same cycle to count as discrete faces.

Similarly, let b be a vertex on a discrete face, δ, in the embedding of H, so δ is of the form

$$b \cdots b.$$

Then the graph obtained by connecting G and H with a new edge, a–b, has a planar embedding whose discrete faces are the union of the discrete faces of G and H, except that faces γ and δ are replaced by one new face

$$a \ldots ab \cdots ba.$$

This is illustrated in Figure 5.43, where the faces of G and H are:

$$G : \{\texttt{axyza}, \texttt{axya}, \texttt{ayza}\} \qquad H : \{\texttt{btuvwb}, \texttt{btvwb}, \texttt{tuvt}\},$$

and after adding the bridge a–b, there is a single connected graph with faces

$$\{\texttt{axyzabtuvwba}, \texttt{axya}, \texttt{ayza}, \texttt{btvwb}, \texttt{tuvt}\}.$$

Does It Work?

Yes! In general, a graph is planar if and only if each of its connected components has a planar embedding as defined in Definition 5.7.2. Unfortunately, proving this fact requires a bunch of mathematics that we don't cover in this text—stuff like geometry and topology. Of course, that is why we went to the trouble of including Definition 5.7.2—we don't want to deal with that stuff in this text and now that we have a recursive definition for planar graphs, we won't need to. That's the good news.

The bad news is that Definition 5.7.2 looks a lot more complicated than the intuitively simple notion of a drawing where edges don't cross. It seems like it would be easier to stick to the simple notion and give proofs using pictures. Perhaps so, but your proofs are more likely to be complete and correct if you work from the discrete Definition 5.7.2 instead of the continuous Definition 5.7.1.

Where Did the Outer Face Go?

Every planar drawing has an immediately-recognizable outer face—its the one that goes to infinity in all directions. But where is the outer face in a planar embedding?

There isn't one! That's because there really isn't any need to distinguish one. In fact, a planar embedding could be drawn with any given face on the outside. An intuitive explanation of this is to think of drawing the embedding on a *sphere* instead of the plane. Then any face can be made the outside face by "puncturing" that face of the sphere, stretching the puncture hole to a circle around the rest of the faces, and flattening the circular drawing onto the plane.

So pictures that show different "outside" boundaries may actually be illustrations of the same planar embedding. For example, the two embeddings shown in Figure 5.44 are really the same.

This is what justifies the "add bridge" case in ~~a planar embedding~~: whatever face is chosen in the embeddings of each of the disjoint planar graphs, we can draw a bridge between them without needing to cross any other edges in the drawing, because we can assume the bridge connects two "outer" faces.

5.7.3 Euler's Formula

The value of the recursive definition is that it provides a powerful technique for proving properties of planar graphs, namely, structural induction. For example, we will now use Definition 5.7.2 and structural induction to establish one of the most basic properties of a connected planar graph; namely, the number of vertices and edges completely determines the number of faces in every possible planar embedding of the graph.

Theorem 5.7.3 (Euler's Formula). *If a connected graph has a planar embedding, then*

$$v - e + f = 2$$

where v is the number of vertices, e is the number of edges, and f is the number of faces.

For example, in Figure 5.38, $|V| = 4$, $|E| = 6$, and $f = 4$. Sure enough, $4 - 6 + 4 = 2$, as Euler's Formula claims.

Proof. The proof is by structural induction on the definition of planar embeddings. Let $P(\mathcal{E})$ be the proposition that $v - e + f = 2$ for an embedding, $\mathcal{E}$.

Base case: ($\mathcal{E}$ is the one-vertex planar embedding). By definition, $v = 1$, $e = 0$, and $f = 1$, so $P(\mathcal{E})$ indeed holds.

Constructor case (split a face): Suppose G is a connected graph with a planar embedding, and suppose a and b are distinct, nonadjacent vertices of G that appear on some discrete face, $\gamma = a \ldots b \cdots a$, of the planar embedding.

Then the graph obtained by adding the edge a–b to the edges of G has a planar embedding with one more face and one more edge than G. So the quantity $v - e + f$ will

remain the same for both graphs, and since by structural induction this quantity is 2 for G's embedding, it's also 2 for the embedding of G with the added edge. So P holds for the constructed embedding.

Constructor case (add bridge): Suppose G and H are connected graphs with planar embeddings and disjoint sets of vertices. Then connecting these two graphs with a bridge merges the two bridged faces into a single face, and leaves all other faces unchanged. So the bridge operation yields a planar embedding of a connected graph with $v_G + v_H$ vertices, $e_G + e_H + 1$ edges, and $f_G + f_H - 1$ faces. Since

$$
\begin{aligned}
&(v_G + v_H) - (e_G + e_H + 1) + (f_G + f_H - 1) \\
&\quad = (v_G - e_G + f_G) + (v_H - e_H + f_H) - 2 \\
&\quad = (2) + (2) - 2 && \text{(by structural induction hypothesis)} \\
&\quad = 2,
\end{aligned}
$$

$v - e + f$ remains equal to 2 for the constructed embedding. That is, P also holds in this case.

This completes the proof of the constructor cases, and the theorem follows by structural induction. ∎

5.7.4 Bounding the Number of Edges in a Planar Graph

Like Euler's formula, the following lemmas follow by structural induction directly from Definition 5.7.2.

Lemma 5.7.4. *In a planar embedding of a connected graph, each edge is traversed once by each of two different faces, or is traversed exactly twice by one face.*

Lemma 5.7.5. *In a planar embedding of a connected graph with at least three vertices, each face is of length at least three.*

Combining Lemmas 5.7.4 and 5.7.5 with Euler's Formula, we can now prove that planar graphs have a limited number of edges:

Theorem 5.7.6. *Suppose a connected planar graph has $v \geq 3$ vertices and e edges. Then*

$$e \leq 3v - 6.$$

Proof. By definition, a connected graph is planar iff it has a planar embedding. So suppose a connected graph with v vertices and e edges has a planar embedding with f faces. By Lemma 5.7.4, every edge is traversed exactly twice by the face boundaries. So the sum of the lengths of the face boundaries is exactly $2e$. Also by Lemma 5.7.5, when $v \geq 3$, each face boundary is of length at least three, so this sum is at least $3f$. This implies that

$$3f \leq 2e. \tag{5.5}$$

But $f = e - v + 2$ by Euler's formula, and substituting into (5.5) gives

$$3(e - v + 2) \leq 2e$$

$$e - 3v + 6 \leq 0$$

$$e \leq 3v - 6$$ ■

5.8 Returning to K_5 and $K_{3,3}$

Theorem 5.7.6 lets us prove that the quadapi can't all shake hands without crossing. Representing quadapi by vertices and the necessary handshakes by edges, we get the complete graph, K_5. Shaking hands without crossing amounts to showing that K_5 is planar. But K_5 is connected, has 5 vertices and 10 edges, and $10 > 3 \cdot 5 - 6$. This violates the condition of Theorem 5.7.6 required for K_5 to be planar, which proves

Corollary 5.8.1. *K_5 is not planar.*

We can also use Euler's Formula to show that $K_{3,3}$ is not planar. The proof is similar to that of Theorem 5.7.6 except that we use the additional fact that $K_{3,3}$ is a bipartite graph.

Lemma 5.8.2. *Every closed walk in a bipartite graph has even length.*

Proof. A bipartite graph $G = (V, E)$ is defined by the property that the nodes V are partitioned into two sets L and R where every edge connects a node in L to a node in R. Hence, any closed walk in G must alternate between a node in L followed by a node in R. Since a closed walk ends on the same node it started with, it must visit nodes in L equally often as it visits nodes in R. Hence it must have even length. ■

Corollary 5.8.3. *In a planar embedding of a connected* bipartite *graph with at least 3 vertices, each face has length at least 4.*

Proof. By Lemma 5.7.5, every face has length 3. Since the graph is bipartite and since

each face is a closed walk, Lemma 5.8.2 implies that no face can have length 3. Hence, every face must have length at least 4. ■

Theorem 5.8.4. *$K_{3,3}$ is not planar.*

Proof. By contradiction. Assume $K_{3,3}$ is planar and consider any planar embedding of $K_{3,3}$ with f faces. Arguing as in the proof of Theorem 5.7.6 (???) (but using Corollary 5.8.3 in place of Lemma 5.7.5 since $K_{3,3}$ is bipartite), we find that the sum of the lengths of the face boundaries is exactly $2e$ and at least $4f$. Hence,

$$4f \leq 2e$$

for any bipartite graph.

Since $K_{3,3}$ is bipartite, we know by Theorem XY that $K_{3,3}$ does not contain any closed walks of odd length. By Lemma 5.7.5, every face has length at least 3. This means that every face in any embedding of $K_{3,3}$ must have length at least 4. Plugging this fact into the proof of Theorem 5.7.6,

Plugging in $e = 9$ and $v = 6$ for $K_{3,3}$ in Euler's Formula, we find that

$$f = 2 + e - v$$

$$= 5.$$

But

$$4 \cdot 5 \not\leq 2 \cdot 9,$$

and so we have a contradiction. Hence $K_{3,3}$ must not be planar. ■

5.8.1 Another Characterization for Planar Graphs

We did not choose to pick on K_5 and $K_{3,3}$ because of their application to dogs getting home or quadapi shaking hands. Rather, we selected these graphs as examples because they provide another way to characterize the set of planar graphs, as follows.

Theorem 5.8.5 (Kuratowski). *A graph is not planar if and only if it contains K_5 or $K_{3,3}$ as a*

minor.

Definition 5.8.6. A *minor* of a graph G is a graph that can be obtained by repeatedly[20] deleting vertices, deleting edges, and merging *adjacent* vertices of G. Here *merging* two adjacent vertices, n_1 and n_2 of a graph means deleting the two vertices and then replacing them by a new "merged" vertex, m, adjacent to all the vertices that were adjacent to either of n_1 or n_2, as illustrated in Figure 5.45.

For example, Figure 5.46 illustrates why C_3 is a minor of the graph in Figure 5.46(a). In fact C_3 is a minor of a connected graph G if and only if G is not a tree.

We will not prove Theorem 5.8.5 here, nor will we prove the following handy facts, which are obvious given the definition of a planar drawing from Section 5.7, and which can be proved using the recursive definition of a planar embedding from Section 5.7.2.

Corollary 5.8.7. *Deleting a vertex from a planar graph, along with all its incident edges, leaves*

[20] The three operations can be performed in any order and in any quantities, or not at all.

another planar graph.

Theorem 5.8.8. *Any subgraph of a planar graph is planar.*

Theorem 5.8.9. *Merging two adjacent vertices of a planar graph leaves another planar graph.*

5.8.2 Coloring Planar Graphs

We've covered a lot of ground with planar graphs, but not nearly enough to prove the famous 4-color theorem. But we can get awfully close. Indeed, we have done almost enough work to prove that every planar graph can be colored using only 5 colors. We need only one more lemma:

Lemma 5.8.10. *Every planar graph has a vertex of degree at most five.*

Proof. By contradiction. If every vertex had degree at least 6, then the sum of the vertex degrees is at least $6v$, but since the sum of the vertex degrees equals $2e$, by the Hand-

shake Lemma (Lemma 5.2.1), we have $e \geq 3v$ contradicting the fact that $e \leq 3v - 6 < 3v$ by Theorem 5.7.6. ■

Theorem 5.8.11. *Every planar graph is five-colorable.*

Proof. The proof will be by strong induction on the number, v, of vertices, with induction hypothesis:

Every planar graph with v vertices is five-colorable.

Base cases ($v \leq 5$): immediate.

Inductive case: Suppose G is a planar graph with $v + 1$ vertices. We will describe a five-coloring of G.

First, choose a vertex, g, of G with degree at most 5; Lemma 5.8.10 guarantees there will be such a vertex.

Case 1: ($\deg(g) < 5$): Deleting g from G leaves a graph, H, that is planar by Corollary 5.8.7, and, since H has v vertices, it is five-colorable by induction hypothesis. Now define a five coloring of G as follows: use the five-coloring of H for all the vertices besides g, and assign one of the five colors to g that is not the same as the color assigned to any of its neighbors. Since there are fewer than 5 neighbors, there will always be such a color available for g.

Case 2: ($\deg(g) = 5$): If the five neighbors of g in G were all adjacent to each other, then these five vertices would form a nonplanar subgraph isomorphic to K_5, contradicting Theorem 5.8.8 (since K_5 is not planar). So there must be two neighbors, n_1 and n_2, of g that are not adjacent. Now merge n_1 and g into a new vertex, m. In this new graph, n_2 is adjacent to m, and the graph is planar by Theorem 5.8.9. So we can then merge m and n_2 into a another new vertex, m', resulting in a new

graph, G', which by Theorem 5.8.9 is also planar. Since G' has $v - 1$ vertices, it is five-colorable by the induction hypothesis.

Define a five coloring of G as follows: use the five-coloring of G' for all the vertices besides g, n_1 and n_2. Next assign the color of m' in G' to be the color of the neighbors n_1 and n_2. Since n_1 and n_2 are not adjacent in G, this defines a proper five-coloring of G except for vertex g. But since these two neighbors of g have the same color, the neighbors of g have been colored using fewer than five colors altogether. So complete the five-coloring of G by assigning one of the five colors to g that is not the same as any of the colors assigned to its neighbors.

■

5.8.3 Classifying Polyhedra

The Pythagoreans had two great mathematical secrets, the irrationality of $\sqrt{2}$ and a geometric construct that we're about to rediscover!

A *polyhedron* is a convex, three-dimensional region bounded by a finite number of polygonal faces. If the faces are identical regular polygons and an equal number of polygons meet at each corner, then the polyhedron is *regular*. Three examples of regular polyhedra are shown in Figure 5.47: the tetrahedron, the cube, and the octahedron.

We can determine how many more regular polyhedra there are by thinking about planarity. Suppose we took *any* polyhedron and placed a sphere inside it. Then we could project the polyhedron face boundaries onto the sphere, which would give an image that was a planar graph embedded on the sphere, with the images of the corners of the polyhedron corresponding to vertices of the graph. We've already observed that

embeddings on a sphere are the same as embeddings on the plane, so Euler's formula for planar graphs can help guide our search for regular polyhedra.

For example, planar embeddings of the three polyhedra in Figure 5.47 are shown in Figure 5.48.

Let m be the number of faces that meet at each corner of a polyhedron, and let n be the number of edges on each face. In the corresponding planar graph, there are m edges incident to each of the v vertices. By the Handshake Lemma 5.2.1, we know:

$$mv = 2e.$$

Also, each face is bounded by n edges. Since each edge is on the boundary of two faces, we have:

$$nf = 2e$$

Solving for v and f in these equations and then substituting into Euler's formula gives:

$$\frac{2e}{m} - e + \frac{2e}{n} = 2$$

which simplifies to

$$\frac{1}{m} + \frac{1}{n} = \frac{1}{e} + \frac{1}{2} \tag{5.6}$$

Equation 5.6 places strong restrictions on the structure of a polyhedron. Every nondegenerate polygon has at least 3 sides, so $n \geq 3$. And at least 3 polygons must meet to form a corner, so $m \geq 3$. On the other hand, if either n or m were 6 or more, then the left side of the equation could be at most $1/3 + 1/6 = 1/2$, which is less than the right side. Checking the finitely-many cases that remain turns up only five solutions, as shown in Figure 5.49. For each valid combination of n and m, we can compute the associated number of vertices v, edges e, and faces f. And polyhedra with these properties do actually exist. The largest polyhedron, the dodecahedron, was the other great

mathematical secret of the Pythagorean sect.

The 5 polyhedra in Figure 5.49 are the only possible regular polyhedra. So if you want to put more than 20 geocentric satellites in orbit so that they *uniformly* blanket the globe—tough luck!

5.9 Problems

Class Problems

Homework Problems

Exam Problems

EDITING NOTE:

Move this from section 5.2.2 to problems.

Kill this and put it into the text already

Now we can immediately see how to color a bipartite graph using only two colors: let all the L vertices be black and all the R vertices be white. Conversely, if a graph is 2-colorable, then it is bipartite with L being the vertices of one color and R the vertices of the other color. In other words,

"bipartite" is a synonym for "*2-colorable.*"

The following Lemma gives another useful characterization of bipartite graphs.

Theorem 5.9.1. *A graph is bipartite iff it has no odd-length cycle.*

The proof of Theorem 5.9.1 is left to Problem ??. ■