12 Simple Graphs

Simple graphs model relationships that are *symmetric*, meaning that the relationship is mutual. Examples of such mutual relationships are being married, speaking the same language, not speaking the same language, occurring during overlapping time intervals, or being connected by a conducting wire. They come up in all sorts of applications, including scheduling, constraint satisfaction, computer graphics, and communications, but we’ll start with an application designed to get your attention: we are going to make a professional inquiry into sexual behavior. Specifically, we’ll look at some data about who, on average, 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 people 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 men have on average 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”—raising some questions about the seriousness of their reporting.

Yet again in August, 2007, the New York 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. So, whose numbers do you think are more accurate: the University of Chicago, ABC News, or the National Center?

Don’t answer—this is a trick question designed to trip you up. Using a little graph theory, we’ll explain why none of these findings can be anywhere near the truth.

12.1 Vertex Adjacency and Degrees

Simple graphs are defined in almost the same way as digraphs, except that edges are *undirected*—they connect two vertices without pointing in either direction between the vertices. So instead of a directed edge $\langle v \to w \rangle$ which starts at vertex v and ends at vertex w, a simple graph only has an undirected edge $\langle v—w \rangle$ that connects v and w.

Definition 12.1.1. A *simple graph* G consists of a nonempty set, $V(G)$, called the *vertices* of G, and a set $E(G)$ called the *edges* of G. An element of $V(G)$ is called a *vertex*. An element of $E(G)$ is an *undirected edge* or simply an “edge.” An undirected edge has two vertices $u \neq v$ called its *endpoints*. Such an edge can be represented by the two element set $\{u, v\}$. The notation $\langle u—v \rangle$ denotes this edge.

Both $\langle u—v \rangle$ and $\langle v—u \rangle$ describe the same undirected edge, whose endpoints are u and v.

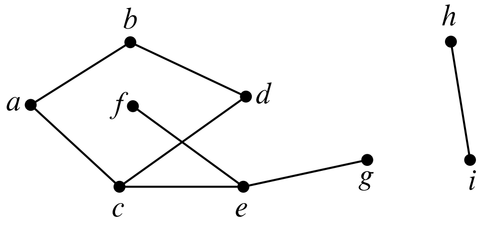


Figure 12.1 An example of a graph with 9 vertices and 8 edges.

For example, let H be the graph pictured in Figure 12.1. The vertices of H correspond to the nine dots in Figure 12.1, that is,

$$V(H) = \{a, b, c, d, e, f, g, h, i\}.$$

The edges correspond to the eight lines, that is,

$$E(H) = \{\, \langle a—b \rangle, \langle a—c \rangle, \langle b—d \rangle, \langle c—d \rangle, \langle c—e \rangle, \langle e—f \rangle, \langle e—g \rangle, \langle h—i \rangle \,\}.$$

Mathematically, that’s all there is to the graph H.

Definition 12.1.2. Two vertices in a simple graph are said to be *adjacent* iff they are the endpoints of the same edge, and an edge is said to be *incident* to each of its endpoints. 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 the number of vertices adjacent to it.

For example, for the graph H of Figure 12.1, vertex a is adjacent to vertex b, and b is adjacent to d. The edge $\langle a—c \rangle$ is incident to its endpoints 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 G does not need to have any edges at all. $|E(G)|$ could be zero, implying that the degree of every vertex would also be zero. But a simple graph must have at least one vertex—$|V(G)|$ is required to be at least one.

Note that in a simple graph there cannot be more than one edge between the same two vertices.[1] Also *self-loops*—edges that begin and at the same vertex—are not allowed in simple graphs.

For the rest of this chapter we'll use "graphs" as an abbreviation for "simple graphs."

A synonym for "vertices" is "*nodes*," and we'll use these words interchangeably. Simple graphs are sometimes called *networks*, edges are sometimes called *arcs* or *lines*. We mention this as a "heads up" in case you look at other graph theory literature; we won't use these words.

12.2 Sexual Demographics in America

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.[2] We'll put an edge between a male and a female iff they have been sexual partners. This graph is pictured in Figure 12.2 with males on the left and females on the right.

Actually, this is a pretty hard graph to figure out, let alone draw. The graph is *enormous*: the US population is about 300 million, so $|V| \approx 300M$. Of these, approximately 50.8% are female and 49.2% are male, so $|M| \approx 147.6M$, and $|F| \approx 152.4M$. 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—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 has exactly one endpoint at an M vertex (remember, we're only considering male-female relationships); so the sum of the degrees of the M vertices equals the number of edges. For the same reason, the sum of the degrees of the F vertices equals the number of edges. So these sums are equal:

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

Now suppose we divide both sides of this equation by the product of the sizes of

[1] *Multigraphs* that have more than one edge with the same two endpoints are sometimes convenient, but they are not needed for the applications we present.

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

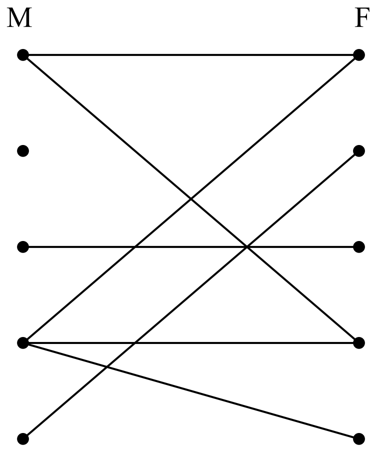


Figure 12.2 The sex partners graph.

the two sets, $|M| \cdot |F|$:

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

The terms above in parentheses are the *average degree of an M vertex* and the *average degree of an F* vertex. So we know:

$$\text{Avg. deg in } M = \frac{|F|}{|M|} \cdot \text{Avg. deg in } F \tag{12.1}$$

In other words, we've proved that the average number of female partners of males in the population compared to the average number of males per female is *determined solely by the relative number of males and females in the population*.

Now the Census Bureau reports that there are slightly more females than males in America; in particular $|F|/|M|$ is about 1.035. So we know that males have on average 3.5% more opposite-gender partners than females, and that this tells us nothing about any sex's promiscuity or selectivity. Rather, it just has to do with 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 couple of 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 students in a minority than students not in that minority, which is, of course, what "minority" means.

12.2.1 Handshaking Lemma

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

Lemma 12.2.1. *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. ■

We refer to Lemma 12.2.1 as the *Handshaking 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.

12.3 Some Common Graphs

Some graphs come up so frequently that they have names. A *complete graph* K_n has n vertices and an edge between every two vertices, for a total of $n(n-1)/2$ edges. For example, K_5 is shown in Figure 12.3.

The *empty graph* has no edges at all. For example, the five-vertex empty graph is shown in Figure 12.4.

An n-vertex graph containing $n-1$ edges in sequence is known as a *line graph* L_n. More formally, L_n has

$$V(L_n) = \{v_1, v_2, \dots, v_n\}$$

and

$$E(L_n) = \{\langle v_1 \text{—} v_2 \rangle, \langle v_2 \text{—} v_3 \rangle, \dots, \langle v_{n-1} \text{—} v_n \rangle\}$$

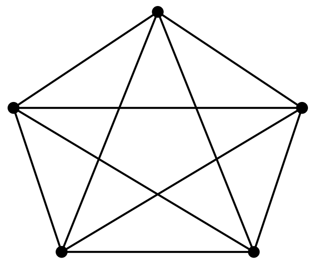


Figure 12.3 K_5: the complete graph on five nodes.

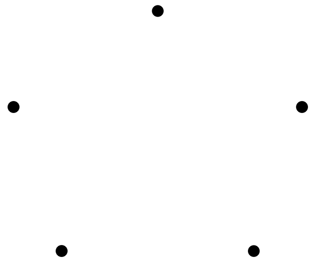


Figure 12.4 An empty graph with five nodes.

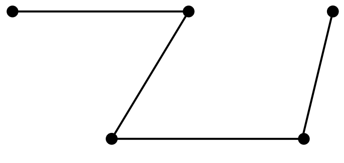


Figure 12.5 L_5: a 5-vertex line graph.

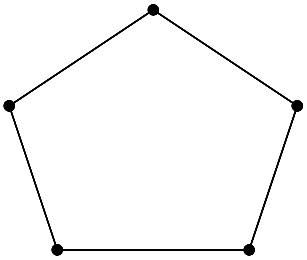


Figure 12.6 C_5: a 5-node cycle graph.

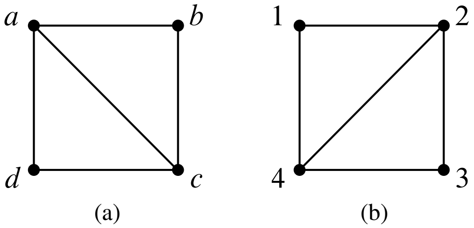


Figure 12.7 Two Isomorphic graphs.

For example, L_5 is pictured in Figure 12.5.

There is also a one-way infinite line graph L_∞ which can be defined by letting the nonnegative integers $\mathbb{N}$ be the vertices with edges $\langle k—(k+1)\rangle$ for all $k \in \mathbb{N}$.

If we add the edge $\langle v_n—v_1\rangle$ to the line graph L_n, we get a graph called a *length-n cycle C_n*. Figure 12.6 shows a picture of length-5 cycle.

12.4 Isomorphism

Two graphs that look different might actually be the same in a formal sense. For example, the two graphs in Figure 12.7 are both four-vertex, five-edge graphs and you get graph (b) by a 90° clockwise rotation of graph (a).

Strictly speaking, these graphs are different mathematical objects, but this difference doesn't reflect the fact that the two graphs can be described by the same picture—except for the labels on the vertices. This idea of having the same picture "up to relabeling" can be captured neatly by adapting Definition 10.7.1 of isomorphism of digraphs to handle simple graphs. An isomorphism between two graphs is an edge-preserving bijection between their sets of vertices:

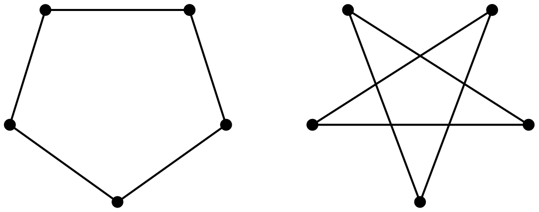


Figure 12.8 Isomorphic C_5 graphs.

Definition 12.4.1. An isomorphism between graphs G and H is a bijection $f : V(G) \to V(H)$ such that

$$\langle u—v \rangle \in E(G) \text{ IFF } \langle f(u)—f(v) \rangle \in E(H)$$

for all $u, v \in V(G)$. Two graphs are isomorphic when there is an isomorphism between them.

Here is an isomorphism f between the two graphs in Figure 12.7:

$$\begin{array}{ll} f(a) ::= 2 & f(b) ::= 3 \\ f(c) ::= 4 & f(d) ::= 1. \end{array}$$

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, Figure 12.8 shows two different ways of drawing C_5.

Notice that if f is an isomorphism between G and H, then f^{-1} is an isomorphism between H and G. Isomorphism is also transitive because the composition of isomorphisms is an isomorphism. In fact, isomorphism is an equivalence relation.

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, since an isomorphism is a bijection between sets of vertices, 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. Thus, v and $f(v)$ will have the same degree. If one graph has a vertex of degree four and another does not, then they can't be isomorphic. In fact, they can't be isomorphic if the number of degree-four 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 guide the search for an isomorphism when there is one. It's generally easy in practice to decide whether two graphs are isomorphic. However, no one has yet found a procedure for determining whether two graphs are isomorphic that is *guaranteed* to run in polynomial time on all pairs of graphs.[3]

Having such a procedure would be useful. For example, it would make 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.

The definitions of bijection and isomorphism apply to infinite graphs as well as finite graphs, as do most of the results in the rest of this chapter. But graph theory focuses mostly on finite graphs, and we will too.

> *In the rest of this chapter we'll assume graphs are finite.*

We've actually been taking isomorphism for granted ever since we wrote "K_n has n vertices…" at the beginning of Section 12.3.

Graph theory is all about properties preserved by isomorphism.

12.5 Bipartite Graphs & Matchings

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: *bipartite graphs*.

Definition 12.5.1. A *bipartite graph* is a graph whose vertices can be partitioned into two blocks, $L(G)$ and $R(G)$, such that every edge has one endpoint in $L(G)$ and the other endpoint in $R(G)$.

So every bipartite graph looks something like the graph in Figure 12.2.

12.5.1 The Bipartite Matching 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. As you might

[3] A procedure runs in *polynomial time* when it needs an amount of time of at most $p(n)$, where n is the total number of vertices and $p()$ is a fixed polynomial.

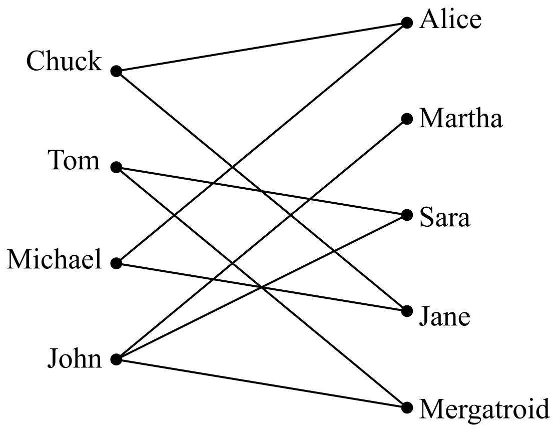


Figure 12.9 A graph where an edge between a man and woman denotes that the man likes the woman.

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 marry every woman to a man so that every man is married at most once.

But what about getting a mate for every man so that every woman is married at most once? 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. In this scenario, the "likes" relationship need not be symmetric, since for the time being, we will only worry about finding a mate for each man that he likes.[4] Later, we will consider the "likes" relationship from the female perspective as well. For example, we might obtain the graph in Figure 12.9.

A *matching* is defined to be an assignment of a woman to each man so that different men are assigned to different women, and a man is always assigned a woman that he likes. For example, one possible matching for the men is shown in Figure 12.10.

[4] It's convenient to describe the matching problem in terms of men and women, since it let's us say things like "a man's wife" instead of "a left hand node's matching right hand node." We don't think marriage must be heterosexual, or that men should be given first choice of mate.

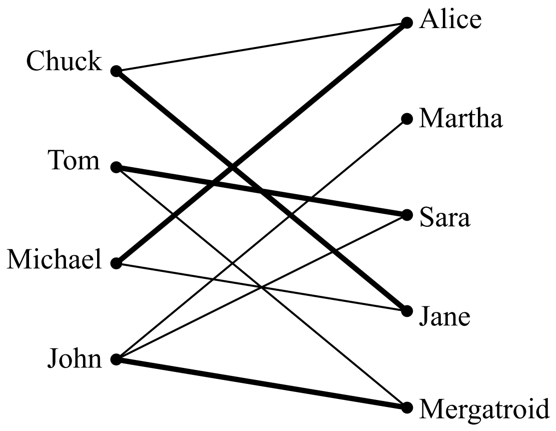


Figure 12.10 One possible matching for the men is shown with bold edges. For example, John is matched with Mergatroid.

12.5.2 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 remarkably 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 in Figure 12.9 consists of Martha, Sara and Mergatroid. For us to have any chance at all of matching up the men, the following *matching condition* must hold:

The Matching Condition: every subset of men likes at least as large a set of women.

For example, we cannot 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 12.5.2. *A matching for a set M of men with a set W of women 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. For any subset of men, each man likes at least the woman he is matched with and a woman is matched with at most one man. 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, on the predicate:

$$\begin{aligned} P(m) ::= {} & \text{if the matching condition holds for a set, } M, \\ & \text{of } m \text{ men, then there is a matching for } M. \end{aligned}$$

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: Suppose that $|M| = m + 1 \geq 2$. To find a matching for M, there are two cases.

Case 1: Every nonempty subset of at most m 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. This leaves m men and one fewer women, and the matching condition will still hold. So the induction hypothesis $P(m)$ implies we can match the remaining m men.

Case 2: Some nonempty subset X of at most m men likes an *equal-size* set Y of women. The matching condition must hold within X, so the strong induction hypothesis implies we can match the men in X with the women in Y. This leaves the problem of matching the set $M - X$ of men to the set $W - Y$ of women.

But the problem of matching $M - X$ against $W - Y$ also satisfies the Matching condition, because any subset of men in $M - X$ who liked fewer women in $W - Y$ would imply there was a set of men who liked fewer women in the whole set W. Namely, if a subset $M_0 \subseteq M - X$ liked only a strictly smaller subset of women $W_0 \subseteq W - Y$, then the set $M_0 \cup X$ of men would like only women in the strictly smaller set $W_0 \cup Y$. So again the strong induction hypothesis implies we can match the men in $M - X$ with the women in $W - Y$, which completes a matching for M.

So in both cases, there is a matching for the men, which completes the proof of the Inductive step. The theorem follows by induction. ■

The proof of Theorem 12.5.2 gives an algorithm for finding a matching in a bipartite graph, albeit not a very efficient one. However, efficient algorithms for finding a matching in a bipartite graph do exist. Thus, if a problem can be reduced to finding a matching, instances of the problem can be solved in a reasonably efficient way.

A Formal Statement

Let's restate Theorem 12.5.2 in mathematical terms so that you'll not always be condemned to saying, "Now this group of men likes at least as many women…"

Definition 12.5.3. A *matching* in a graph G is a set of edges $M \subseteq E(G)$ such that no two edges in M are incident to the same vertex. The vertices incident to an edge in M are said to be *covered* by M. A matching is said to be *perfect* when it covers $V(G)$. The set $\text{N}(S)$ of *neighbors* of some set $S \subseteq V(G)$ of vertices is the image of S under the edge-relation, that is,

$$\text{N}(S) ::= \{ r \mid \langle s\text{—}r \rangle \in E(G) \text{ for some } s \in S \}.$$

S is called a *bottleneck* if

$$|S| > |\text{N}(S)|.$$

Theorem 12.5.4 (Hall's Theorem)**.** *Let G be a bipartite graph. There is a matching in G that covers $L(G)$ iff no subset of $L(G)$ 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. 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 12.5.5. A bipartite graph G is *degree-constrained* when there is an integer $d \geq 1$ such that

$$\deg(l) \geq d \geq \deg(r) \tag{12.2}$$

for every $l \in L(G)$ and $r \in R(G)$.

For example, the graph in Figure 12.9 is degree-constrained with $d = 2$, since every vertex on the left is adjacent to at least two vertices on the right while every vertex on the right is adjacent to at most two vertices on the left.

Theorem 12.5.6. *If G is a degree-constrained bipartite graph, then there is a matching that covers $L(G)$.*

Proof. Suppose G satisfies (12.2), and let S be a subset of $L(G)$. We will show that S satisfies Hall's condition, namely,

$$|\operatorname{N}(S)| \geq |S|. \tag{12.3}$$

Let D_S be the set of edges with an endpoint in a S. According to (12.2), every vertex in S is the endpoint of at least d edges,

$$|D_S| \geq d|S|.$$

Likewise, any edge in D_S has an endpoint in $\operatorname{N}(S)$ by definition. According to (12.2), every vertex in $\operatorname{N}(S)$ is incident to at most d edges, so we also have

$$d|\operatorname{N}(S)| \geq |D_S|.$$

Combining the above inequalities gives

$$d|\operatorname{N}(S)| \geq d|S|.$$

Now cancelling d completes the derivation of equation (12.3). ■

A graph is *regular* when all its vertice have the same degree. Regular graphs come up in lots of examples. Since a nonempty bipartite regular graph is degree-constrained by definition, Theorem 12.5.6 guarantees it will have a perfect matching.

12.6 Coloring

In Section 12.2, we used edges to indicate an affinity between a pair of nodes. But there are lots of situations in which edges will correspond to *conflicts* between nodes. Exam scheduling is a typical example.

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

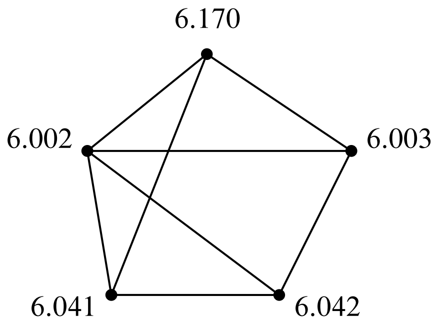


Figure 12.11 A scheduling graph for five exams. Exams connected by an edge cannot be given at the same time.

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 12.11.

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 12.12, three colors suffice for our example.

The coloring in Figure 12.12 corresponds to giving one final on Monday morning (red), 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 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 12.6.1. The minimum number of colors for which a graph G has a valid

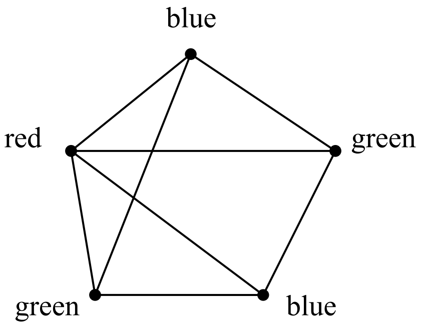


Figure 12.12 A 3-coloring of the exam graph from Figure 12.11.

coloring is called its *chromatic number*, $\chi(G)$.

So G is k-colorable iff $\chi(G) \leq k$.

12.6.2 Some Coloring Bounds

There are some simple properties of graphs that give useful bounds on colorability.

The simplest property is being a cycle: an even-length cycle is 2-colorable. So

$$\chi(C_{\text{even}}) = 2.$$

On the other hand, an odd-length cycle requires 3 colors, that is,

$$\chi(C_{\text{odd}}) = 3. \tag{12.4}$$

You should take a moment to think about why this equality holds.

Another simple example is a complete graph K_n:

$$\chi(K_n) = n$$

since no two vertices can have the same color.

Being bipartite is another property closely related to colorability. If a graph is bipartite, then you can color it with 2 colors using one color for the nodes on the “left” and a second color for the nodes on the “right.” Conversely, graphs with chromatic number 2 are all bipartite with all the vertices of one color on the “left” and those with the other color on the right. Since only graphs with no edges—the *empty graphs*—have chromatic number 1, we have:

Lemma 12.6.2. *A nonempty graph G is bipartite iff $\chi(G) = 2$.*

The chromatic number of a graph can also be shown to be small if the vertex degrees of the graph are small. In particular, 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 12.6.3. *A graph with maximum degree at most n is* $(n+1)$*-colorable.*

Now you might be tempted to try to prove this theorem using induction on n, but this would be a poor choice—we actually don't know any way to this work, so we expect it would ruin your week if this was your approach on a problem set. On the other hand, the number of vertices or the number of edges are typically good quantities on which to base induction proofs about graphs. We will prove Theorem 12.6.3 by induction on the number of vertices.

Proof of Theorem 12.6.3. Let $n \in \mathbb{N}$ be fixed. The induction hypothesis will be:

$$P(k) ::= \text{If a } k\text{-vertex graph has maximum degree } n\text{, then it is } (n+1)\text{-colorable.}$$

.

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

Inductive step: Now assume that $P(k)$ is true, and let G be an $(k+1)$-vertex graph with maximum degree at most n. Remove a vertex v and all edges incident to v from G, leaving a k-vertex subgraph H. The maximum degree of H is still at most n, and so we can find an $(n+1)$-coloring for H by induction hypothesis.

Now add back vertex v. Since there are more colors—$n+1$—than vertices adjacent to v—at most $\deg(v) \le n$—we can now pick a color for v that is different from the colors of all these adjacent vertices. This yields a coloring of G with $(n+1)$-colors. So we have shown that G is $(n+1)$-colorable, as required. ■

Sometimes $n+1$ colors is the best you can do. For example, every node in K_{n+1} has degree n and $\chi(K_{n+1}) = n+1$, so this is an example where Theorem 12.6.3 gives the best possible bound. Theorem 12.6.3 also gives the best possible coloring bound for *any* graph with degree bounded by n that has K_{n+1} as a subgraph.

But sometimes $n+1$ colors are far from the best that you can do. For example, the *star graph* shown in Figure 12.13 has maximum degree six but can be colored using just two colors.

In general it is very hard to find a minimal coloring for an arbitrary graph. In fact, just calculating the chromatic number is hard. We'll say more about this in the next section.

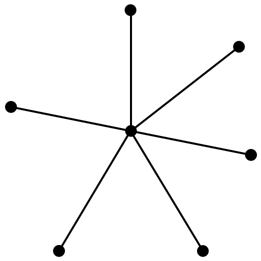


Figure 12.13 A seven-vertex star graph.

12.6.3 Why coloring?

One reason coloring problems frequently arise in practice is because scheduling conflicts are so common. For example, at the internet company Akamai, cofounded by Tom Leighton, a new version of software is deployed over each of its servers (200,000 servers in 2016) every few days. It would take more than twenty years to update all these the servers one at a time, so the deployment must be carried out for many servers simultaneouly. On the other hand, certain pairs of servers with common critical functions cannot be updated simultaneouly, since a server needs to be taken offline while being updated.

This problem gets solved by making a 200,000-node conflict graph and coloring it with with a dozen or so colors—so only a dozen or so waves of installs 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 it is important 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 1.1.4. How many colors are needed to color a map so that adjacent territories get different colors? Answering the question has no direct practical applications, but it tantalized graph theorists for over a century until an answer was discovered—to great acclaim—in the 1970's: four colors are enough to color any map. A reasonably efficient way to find an actual 4-coloring can be based on that proof.

Surprisingly, the question whether there is an efficient procedure to tell if an arbitrary planar graph really *needs* four colors—or if three would do the job—turns out to be the SAT problem of Section 3.5 in disguise. So you win a million dollars if you figure out how hard it is to determine 3-colorability. A proof that testing 3-colorability of planar graphs is as hard as SAT is given in Problems 12.31 and 12.32.

12.7 Walks in Simple Graphs

12.7.1 Walks, Paths, Cycles

Walks and paths in simple graphs are esentially the same as in digraphs. We just modify the digraph definitions using undirected edges instead of directed ones. For example, the formal definition of a walk in a simple graph is a virtually the same as the Definition 10.2.1 of a walk in a digraph:

Definition 12.7.1. A *walk in a simple graph G* is an alternating sequence of vertices and edges that begins with a vertex, ends with a vertex, and such that for every edge $\langle u—v \rangle$ in the walk, one of the endpoints u, v is the element just before the edge, and the other endpoint is the next element after the edge. The *length of a walk* is the total number of occurrences of edges in it.

So a walk $\mathbf{v}$ is a sequence of the form

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

where $\langle v_i—v_{i+1} \rangle \in E(G)$ for $i \in [0..k)$. The walk is said to *start* at v_0, to *end* at v_k, and the *length*, $|\mathbf{v}|$, of the walk is k. The walk is a *path* iff all the v_i's are different, that is, if $i \neq j$, then $v_i \neq v_j$.

A walk that begins and ends at the same vertex is a *closed walk*. A single vertex counts as a length zero closed walk as well as a length zero path.

A *cycle* can be represented by a closed walk of length three or more whose vertices are distinct except for the beginning and end vertices.

Note that in contrast to digraphs, we don't count length two closed walks as cycles in simple graphs. That's because a walk going back and forth on the same edge is neither interesting nor important. There are also no closed walks of length one, since simple graphs don't have self-loops.

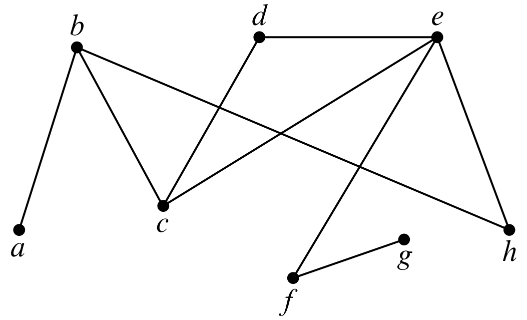


Figure 12.14 *A graph with 3 cycles: $bhecb$, $cdec$, $bcdehb$.*

As in digraphs, the length of a walk is the number of occurrences of *edges* in the walk, which is *one less* than the number of occurrences of vertices. For example, the graph in Figure 12.14 has a length 6 path through the seven successive vertices $abcdefg$. This is the longest path in the graph. The graph in Figure 12.14 also has three cycles through successive vertices $bhecb$, $cdec$ and $bcdehb$.

12.7.2 Cycles as Subgraphs

We don’t intend that cycles have a beginning or an end, so *any* of the paths that go around a cycle can represent it. For example, in the graph in Figure 12.14, the cycle starting at b and going through vertices $bcdehb$ can also be described as starting at d and going through $dehbcd$. Furthermore, cycles in simple graphs don’t have a direction: $dcbhed$ describes the same cycle as though it started and ended at d but went in the opposite direction.

A precise way to explain which closed walks represent the same cycle is to define a cycle to be *subgraph* isomorphic to a cycle graph C_n.

Definition 12.7.2. A graph G is said to be a *subgraph* of a graph H if $V(G) \subseteq V(H)$ and $E(G) \subseteq E(H)$.

For example, the one-edge graph G where

$$V(G) = \{g, h, i\} \quad \text{and} \quad E(G) = \{\langle h—i \rangle\}$$

is a subgraph of the graph H in Figure 12.1. On the other hand, any graph containing an edge $\langle g—h \rangle$ will not be a subgraph of H because this edge is not in $E(H)$. Another example is an empty graph on n nodes, which will be a subgraph of an L_n with the same set of vertices; similarly, L_n is a subgraph of C_n, and C_n is a subgraph of K_n.

Definition 12.7.3. For $n \geq 3$, let C_n be the graph with vertices $[0..n)$ and edges

$$\langle 0—1 \rangle\,,\ \langle 1—2 \rangle\,,\ \ldots\,,\ \langle (n-2)—n-1 \rangle\,,\ \langle n-1—0 \rangle\,.$$

A *cycle of a graph G* is a subgraph of G that is isomorphic to C_n for some $n \geq 3$.

This definition formally captures the idea that cycles don't have direction or beginnings or ends.

12.8 Connectivity

Definition 12.8.1. *Two vertices are connected* in a graph when there is a path that begins at one and ends at the other. By convention, every vertex is connected to itself by a path of length zero. A *graph is connected* when every pair of vertices are connected.

12.8.1 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 American 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.

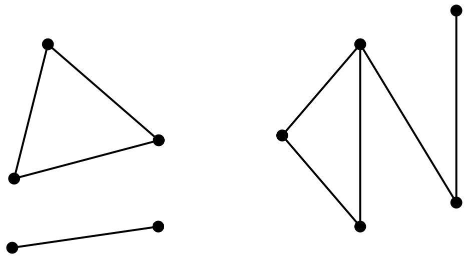


Figure 12.15 One graph with 3 connected components.

Another example is shown in Figure 12.15, which looks like a picture of three graphs, but is intended to be a picture of *one* graph. This graph consists of three pieces. Each piece is a subgraph that 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 12.8.2. A *connected component* of a graph is a subgraph 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. At the other extreme, the empty graph on n vertices has n connected components, each consisting of a single vertex.

12.8.2 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, when the graph is 2-colorable.

Theorem 12.8.3. *The following graph properties are equivalent:*

1. *The graph contains an odd length cycle.*
2. *The graph is not 2-colorable.*
3. *The graph contains an odd length closed walk.*

In other words, if a graph has any one of the three properties above, then it has all of the properties.

We will show the following implications among these properties:

1. IMPLIES 2. IMPLIES 3. IMPLIES 1.

So each of these properties implies the other two, which means they all are equivalent.

1. IMPLIES 2. *Proof.* This follows from equation 12.4. ■

2. IMPLIES 3. If we prove this implication for connected graphs, then it will hold for an arbitrary graph because it will hold for each connected component. So we can assume that G is connected.

Proof. Pick an arbitrary vertex r of G. Since G is connected, for every node $u \in V(G)$, there will be a walk $\mathbf{w}_u$ starting at u and ending at r. Assign colors to vertices of G as follows:

$$\text{color}(u) = \begin{cases} \text{black}, & \text{if } |\mathbf{w}_u| \text{ is even,} \\ \text{white}, & \text{otherwise.} \end{cases}$$

Now since G is not colorable, this can't be a valid coloring. So there must be an edge between two nodes u and v with the same color. But in that case

$$\mathbf{w}_u \,\widehat{}\, \text{reverse}(\mathbf{w}_v) \,\widehat{}\, \langle v—u \rangle$$

is a closed walk starting and ending at u, and its length is

$$|\mathbf{w}_u| + |\mathbf{w}_v| + 1$$

which is odd. ■

3. **IMPLIES** 1. *Proof.* Since there is an odd length closed walk, the WOP implies there is an odd length closed walk $\mathbf{w}$ of minimum length. We claim $\mathbf{w}$ must be a cycle. To show this, assume to the contrary that $\mathbf{w}$ is not a cycle, so there is a repeat vertex occurrence besides the start and end. There are then two cases to consider depending on whether the additional repeat is different from, or the same as, the start vertex.

In the first case, the start vertex has an extra occurrence. That is,

$$\mathbf{w} = \mathbf{f}\,\widehat{x}\,\mathbf{r}$$

for some positive length walks $\mathbf{f}$ and $\mathbf{r}$ that begin and end at x. Since

$$|\mathbf{w}| = |\mathbf{f}| + |\mathbf{r}|$$

is odd, exactly one of $\mathbf{f}$ and $\mathbf{r}$ must have odd length, and that one will be an odd length closed walk shorter than $\mathbf{w}$, a contradiction.

In the second case,

$$\mathbf{w} = \mathbf{f}\,\widehat{y}\,\mathbf{g}\,\widehat{y}\,\mathbf{r}$$

where $\mathbf{f}$ is a walk from x to y for some $y \neq x$, and $\mathbf{r}$ is a walk from y to x, and $|\mathbf{g}| > 0$. Now $\mathbf{g}$ cannot have odd length or it would be an odd-length closed walk shorter than $\mathbf{w}$. So $\mathbf{g}$ has even length. That implies that $\mathbf{f}\,\widehat{y}\,\mathbf{r}$ must be an odd-length closed walk shorter than $\mathbf{w}$, again a contradiction.

This completes the proof of Theorem 12.8.3. ■

12.9 Special Walks and Tours

12.9.1 Euler Tours

Can you walk every hallway in the Museum of Fine Arts *exactly once*? If we represent hallways with vertices and connections between hallways with edges, then

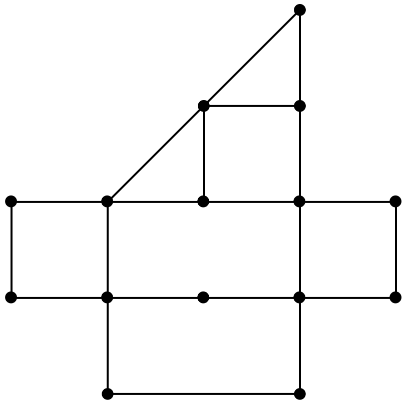


Figure 12.16 Is there a walk that includes every edge exactly once?

this reduces to a question about graphs. For example, could you visit every hallway exactly once in a museum with floor plan represented by the graph in Figure 12.16?

The entire field of graph theory began when the famous 17th century mathematician Leonhard Euler[5] asked whether the seven bridges of Königsberg could all be crossed exactly once—essentially the same question we asked about the Museum of Fine Arts.

An *Euler tour*[6] of a graph is a closed walk that includes every edge exactly once. Graphs with Euler tours turn out to be easy to recognize:

Theorem 12.9.1. *A connected graph has an Euler tour if and only if every vertex has even degree.*

The proof is worked out in a series of easy steps in Problem 12.44. Implicit in the proof is a simple way to find Euler tours when they exist, namely, grow a tour by repeatedly splicing in closed walks until all the edges are included.

Similarly, there is a walk that includes every edge exactly once but is *not* a tour—it begins and ends at different verticies—if and only if there are exactly two vertices of odd degree. The graph shown in Figure 12.16 has exactly two vertices with odd degree, so it has a such a walk but not an Euler tour.

12.9.2 Hamiltonian Cycles

Hamiltonian cycles are the unruly cousins of Euler tours.

[5] Same Euler who defined the constant $e \approx 2.718$ and the totient function ϕ—he did a lot of stuff.

[6] In some other texts, Euler “tours” are called Euler “circuits.”

Definition 12.9.2. A *Hamiltonian cycle* in a graph G is a cycle that visits every *vertex* in G exactly once. Similarly, a *Hamiltonian* path is a path in G that visits every vertex 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 or a Hamiltonian path can be a lot harder than finding an Euler tour or path. No one has yet discovered a simple characterization for when a graph has 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 3.5: you get a million dollars from the Clay Institute for finding an efficient way to determine whether a graph has a Hamiltonian cycle—or for proving that there is no procedure that works efficiently on all graphs.

12.9.3 The Traveling Salesperson Problem

In many applications, there are numerical costs or weights associated with the edges of the graph. For example, suppose the nodes of a graph represent buildings and edges represent connections between them. The cost of a connection may vary a lot from one pair of buildings or towns to another. Another example is where the nodes represent cities, and the weight of an edge is the distance between them: the weight of the Los Angeles/New York City edge is much higher than the weight of the NYC/Boston edge.

As if the problem of finding a Hamiltonian cycle is not hard enough, in a weighted graph, we sometimes need to find a Hamiltonian cycle with *minimum possible sum* of the weights of its edges. This is the famous *Traveling Salesperson Problem*.

For example, suppose you would like to visit every node in the graph shown in Figure 12.17 exactly once, finishing at the node where you started. Can you find a weight 15 cycle that does this?

12.10 *k*-connected Graphs

If we think of a graph as modeling 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. One measure of connection strength is how many links must fail before connectedness fails. In particular, two vertices are *k-edge connected* when it takes at least k “edge-failures” to disconnect them. More precisely:

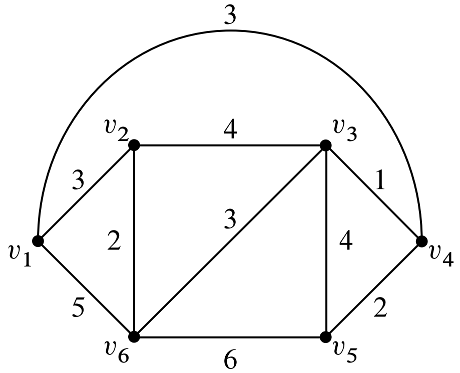


Figure 12.17 Is there a Hamiltonian cycle with weight 15?

Definition 12.10.1. Two vertices in a graph are *k-edge connected* when they remain connected in every subgraph obtained by deleting up to $k-1$ edges. A graph with two or more vertices is k-edge connected when every pair of distinct vertices in the graph are k-edge connected.

From now on we'll drop the "edge" modifier and just say "k-connected."[7]

For example, in the graph in figure 12.14, vertices c and e are 3-connected, b and e are 2-connected, g and e are 1 connected, and no vertices are 4-connected. The graph as a whole is only 1-connected. A complete graph K_n is $(n-1)$-connected. Every cycle is 2-connected.

Notice that two vertices are 1-connected iff they are connected according to Definition 12.8.1.

The idea of a *cut edge* is a useful way to explain 2-connectivity.

Definition 12.10.2. If two vertices are connected in a graph G, but not connected when an edge e is removed, then e is called a *cut edge* of G.

So a graph with more than one vertex is 2-connected iff it is connected and has no cut edges. The following Lemma is a more or less another immediate consequence of the definition of cut edge.

Lemma 12.10.3. *An edge is a cut edge iff it is not on a cycle.*

Proof. An edge on a cycle is clearly not a cut edge. Conversely, if an edge is not a cut edge, then even after it is removed, there must still be a path between its endpoints. Then this path together with the edge form a cycle. ■

[7] There is a corresponding definition of k-vertex connectedness based on deleting vertices rather than edges. Graph theory texts usually use "k-connected" as shorthand for "k-vertex connected." But edge-connectedness will be enough for us.

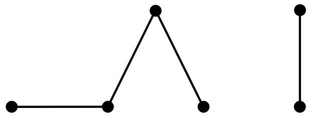


Figure 12.18 A 6-node forest consisting of 2 component trees.

More generally, if two vertices are connected by k edge-disjoint paths—that is, no edge occurs in two paths—then they must be k-connected, since at least one edge will have to be removed from each of the paths before they could disconnect. 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-connected, then there are k edge-disjoint paths connecting them. It takes some ingenuity to prove this just for the case $k = 2$.

12.11 Forests & Trees

We've already made good use of acyclic digraphs, but acyclic *simple* graphs are arguably the most important graphs in computer science.

Definition 12.11.1. An acyclic graph is called a *forest*. A connected acyclic graph is called a *tree*.

The graph shown in Figure 12.18 is a forest. Each of its connected components is by definition a tree.

12.11.1 Leaves

One of the first things you will notice about forests is that they tend to have a lot of vertices with degree one.

Definition 12.11.2. A vertex of degree at most one in a forest is called a *leaf*. A vertex of degree zero is called an *isolated vertex*.

The forest in Figure 12.18 has 4 leaves. The tree in Figure 12.19 has 5 leaves.

A forest with an isolated vertex won't be connected unless it consists of just the one vertex. So if a *connected* forest—that is, a tree—has at least two vertices, then its leaves will be the degree one vertices. Trees don't necessarily have a lot of leaves: the line graph L_n has only two leaves. But two leaves is the minimum.

Lemma 12.11.3. *If a finite tree has more than one vertex, then it has at least two leaves of degree one.*

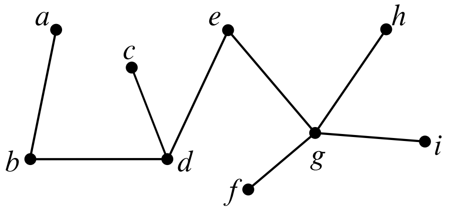


Figure 12.19 A 9-vertex tree with 5 leaves.

Proof. Let u be a vertex at one end of a *longest* path in the tree. If there was an edge between u and some vertex not on the path, then the path could be lengthened by adding that vertex to the path, so there can't be such an edge.

Since the tree has more than one vertex, a longest path must include at least one edge, so there will be an final edge $\langle v\text{—}u\rangle$ at the end of the path. Now if there was an edge from u to a vertex $w \neq v$ on the path, then the part of the path from w to u followed by the edge back to w would be a cycle. So $\langle v\text{—}u\rangle$ must be the only edge incident to u. That is, u is a degree one leaf. Likewise, the vertex at the other end of the path must also be a leaf. ■

12.11.2 Tree Properties

There are many equivalent ways to characterize trees, which helps explain why they come up so often.

Theorem 12.11.4. *For any simple graph G the following properties are equivalent:*

(1) connected and acyclic (G is a tree*).*

(2) connected and every edge is a cut edge (G is edge-minimal connected*).*

(3) acyclic and adding an edge between any two nonadjacent vertices creates a cycle (G is edge-maximal acyclic*).*

(4) a unique path between every pair of vertices.

Proof. We prove (4) implies (2) implies (1) implies (3) implies (4). Expect for the last one, these implications are straightforward.

- (4) implies (2): Connectedness is immediate from (4).

  Supppose $\langle u\text{—}v\rangle$ is an edge of G. Then the unique path between u and v must be just this edge. Removing this edge therefore leaves no path between u and v, which implies that the graph with $\langle u\text{—}v\rangle$ removed is not connected.

- (2) implies (1): Connectedness is immediate from (2). Since an edge on a cycle is not a cut edge (Lemma 12.10.3), the graph must be acyclic.
- (1) implies (3): Connectedness implies there is a path between any two distinct vertices u and v. If u and v are not adjacent, then an additional edge $\langle u—v \rangle$ together with the path between forms a cycle.
- (3) implies (4): Call two vertices a *different-path-pair* (dpp) if there are distinct paths between them. (Note that neither of the vertices forming a dpp are necessarily on a cycle, see Problem 12.35.) Assume for the sake of contradiction that there is a dpp.

  Let $u \neq v$ be a dpp such that some path $\mathbf{p}$ between u and v has *minimum length* among all paths between the vertices in any dpp. Since u and v are a dpp, there must be another path besides $\mathbf{p}$ between them.

  Suppose the other path shares with $\mathbf{p}$ a vertex w different from the endpoints u and v. Then either u and w, or w and v, must be a dpp. But the shortest path between u and w as well as the shortest path between w and v are each shorter than $\mathbf{p}$, contradicting minimality. So the two paths between u and v must not share any vertex other than their endpoints. This means the two paths form a cycle, contradicting (3).

■

The equivalences of Theorem 12.11.4 hold even for infinite trees.[8] For finite trees, there are two further helpful characterizations of trees by numbers of vertices and edges.

Theorem 12.11.5. *A finite forest F consists of exactly $|V(F)| - |E(F)|$ trees.*

Proof. By induction on the number n of vertices. The induction hypothesis will be

$$P(n) ::= \text{every forest } F \text{ with } n \text{ vertices consists of } n - |E(F)| \text{ trees.}$$

Base case: ($n = 1$). In this case, F consists of a tree that is a single isolated vertex and $|E(F)| = 0$. So $P(1)$ is immediate.

Inductive step: ($|V(F)| = n + 1$).

Suppose F has an isolated vertex v. Then removing v yields a graph $F - v$ which will be a forest with $n \geq 1$ vertices. By induction hypothesis, $F - v$ consists

[8] We encourage you to check that the proofs made no use of finiteness.

of $n - |E(F - v)| = n - |E(F)|$ trees. So F consists of these trees along with v, and therefore F consists of $(n + 1) - |E(F)|$ trees, which proves $P(n + 1)$.

Otherwise, every component of F has at least two vertices. Let T be a component tree of F. By Lemma 12.11.3, T has a leaf v of degree one. Let $F - v$ be the result of deleting v and its incident edge. Since deleting vertices and edges will not create a cycle, $F - v$ is a forest with n vertices. In fact the component trees of $F - v$ are the same as those of F with T replaced by $T - v$. In particular, F and $F - v$ have the same number of component trees.

By induction hypothesis, $F - v$ consists of

$$\begin{aligned} n - |E(F - v)| &= n - (|E(F)| - 1) \\ &= (n + 1) - |E(F)| \end{aligned}$$

trees. So F also consists of $(n + 1) - |E(F)|$ component trees, which proves $P(n + 1)$. ■

Theorem 12.11.6. *For any simple graph G the following properties are equivalent:*

(1) finite tree.

(2) finite forest with $|V(G)| = |E(G)| + 1$.

(3) finite, connected and $|V(G)| = |E(G)| + 1$.

Proof. We prove that (1) implies (2) implies (3) implies (1).

- (1) implies (2): Since a finite tree is a finite forest with one component, Theorem 12.11.5 immediately implies (2).

- (2) implies (3): By Theorem 12.11.5, a finite finite forest has $|V(G)| - |E(G)| = 1$ component and is therefore connected by definition of component.

- (3) implies (1): Suppose G is finite and connected. Then there is a set $E \subseteq E(G)$ of edges on cycles of G such that removing these edges leaves a connected acyclic graph $G - E$. Since $G - E$ has one component, Theorem 12.11.5 implies that

$$\begin{aligned} 1 =& |V(G - E)| - |E(G - E)| \\ &= |V(G)| - (|E(G)| - |E|) \\ &= (|V(G)| - |E(G)|) + |E| \\ &= 1 + |E| && \text{(by (3))}. \end{aligned}$$

So $|E| = 0$, which means that G must have been acyclic to begin with.

■

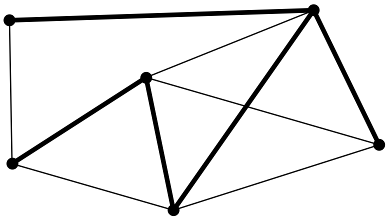


Figure 12.20 A graph where the edges of a spanning tree have been thickened.

12.11.3 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 called a *spanning tree* for the graph. For example, Figure 12.20 is a connected graph with a spanning tree highlighted.

Definition 12.11.7. A *spanning subgraph* of a graph G is a subgraph whose vertices are the same as the vertices of G.

Theorem 12.11.8. *Every finite connected graph contains a spanning tree.*

Proof. Suppose G is a finite connected graph, so the graph G itself is a connected, spanning subgraph. So by WOP, G must have a edge-minimal connected, spanning subgraph T, which by Theorem 12.11.4.2 will be a tree. ■

12.11.4 Minimum-Weight Spanning Trees

Spanning trees are particularly interesting and important for *weighted graphs*—graphs whose edges are assigned numerical weights. The overall *weight* of a weighted graph is the sum of the weights of all its edges. For example, the weight of the spanning tree shown in Figure 12.21(a) is 19.

The general goal is to find a spanning tree with minimum weight, called a *minimum-weight spanning tree* (*MST*).

The spanning tree shown in Figure 12.21(a) turns out *not* to be an MST of the graph in Figure 12.21(b): Figure 12.22 shows another spanning tree for this graph with weight 17. This weight 17 turns out to be an MST, but how do we prove it? In general, how do we find an MST for a connected graph G? We could try enumerating all subtrees of G, but that approach would be hopeless for large graphs.

For the rest of this section, suppose G is a *connected* weighted graph. We'll now explain a simple, general way to find an MST of G.

Definition 12.11.9. A *black-white coloring* of a simple graph G is a partition of the vertices $V(G)$ into two blocks—the "black" vertices and the "white" vertices. A *gray edge* of the coloring is an edge of G with different colored endpoints.

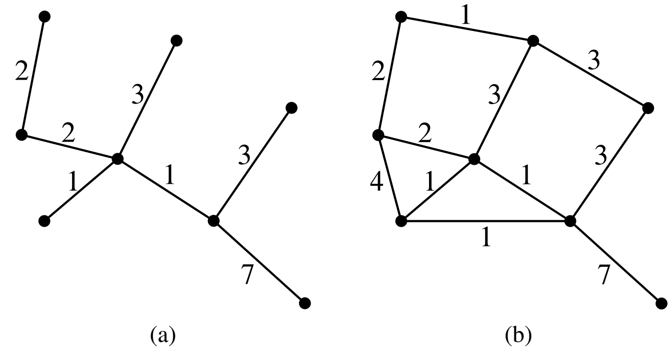


Figure 12.21 A spanning tree (a) with weight 19 for a graph (b).

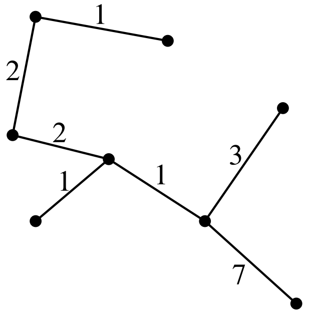


Figure 12.22 An MST with weight 17 for the graph in Figure 12.21(b).

Lemma 12.11.10. *If G is a connected graph, then every black-white coloring of G contains a gray edge.*

Proof. Any coloring of G must have at least one vertex of each color—this follows because the blocks of a partition are by definition nonempty.

Since G is connected, there will be a path that begins at a black vertex and ends at a white vertex. Therefore at some point on the path there has to be an edge that starts with a black vertex and ends with a white one. This is the gray edge. ■

The weighted graphs examples up to now each had two or more edges with the same weight. This is common. But our story becomes simpler if all weights are different.

> *For the rest of this section, assume there are no same-weight edges: different edges have different weights.*

There is really no loss of generality in assuming different weights, since everything we do below assuming distinct weights would work if we assigned an arbitrary rank to same-weight edges and treated higher rank vertices as having larger weight.

The following lemma now makes it easy to find MST's.

Lemma 12.11.11 (Gray Edge Lemma)**.** *For any black-white coloring of a connected graph G, the minimum-weight edge among the gray edges of the coloring must be an edge of* every *MST of G.*

So to find an MST for G, we just need to find enough minimum-weight gray edges to form a spanning tree. We do this using:

Minimum-Weight Gray Edge Procedure

> Start off with the *disconnected* spanning forest consisting of the vertices of G with all edges removed. Next, suppose S is a spanning forest built up at some point by the procedure. If S is not yet connected, then pick a black-white coloring of G such that such that all the vertices in the same connected component of S have the same color. Gray edges now connect different components of S, so none of them are in S. Update S by adding the minimum weight gray edge of this coloring to the edges of S. Continue until S becomes connected.

Theorem 12.11.12. *The Minimum-Weight Gray Edge Procedure will terminate with an MST.*

Proof. To see why this procedure works, suppose G has n vertices and let S be the current spanning forest at any point in the procedure. If S is not connected, a black-white coloring is chosen so that the endpoints of any edge of S are the same color. This ensures that no gray edge is an edge of S.

By Lemma 12.11.10, there must be at least one gray edge in the coloring, so there will be a minimum-weight gray edge.[9] Updating S by adding this gray edge decreases the number of connected components of S by one, so after adding $n - 1$ gray edges, the procedure will terminate with a connected spanning forest, that is, a spanning *tree* T.

By the Gray Edge Lemma 12.11.11, every edge of T has to be in every MST of G. This is only possible if all MST's have the same edges as T, that is, they all equal T. So T is the unique possible MST of G. ■

Corollary 12.11.13. *Every connected graph with no same-weight edges has a unique MST.*

Here are three known algorithms that are special cases of the Minimum Weight Gray Edge Procedure;

Algorithm 1. *[Prim] Grow S one edge at a time by adding a minimum weight edge among the edges that have exactly one endpoint in S.*

This is the algorithm that comes from coloring the growing tree white and all the vertices not in the tree black. Then the gray edges are the ones with exactly one endpoint in the tree.

Algorithm 2. *[Kruskal] Grow S one edge at a time by adding a minimum weight edge among the edges that do not create a cycle in S.*

An edge does not create a cycle in S iff it connects different components of S. So the edge chosen by Kruskal's algorithm will be the minimum weight gray edge when the components it connects are assigned different colors.

The coloring that explains Algorithm 1 also justifies a more flexible algorithm which has Algorithm 1 as a special case:

Algorithm 3. *Grow a forest one edge at a time by picking any component and adding a minimum weight edge among the edges leaving that component.*

This algorithm allows components that are not too close to grow in parallel and independently, which is great for "distributed" computation where separate processors share the work with limited communication between processors.[10]

[9] . . . by the Well Ordering Principle for finite sets Lemma2.4.5.

[10] The idea of growing trees in parallel seems first to have been developed in by Borůvka (1926), ref TBA. Efficient MST algorithms running in parallel time $O(\log |V|)$ are described in Karger, Klein, and Tarjan (1995), ref TBA.

Ok, to wrap up this story, all that's left is the proof of Lemma 12.11.11 that min-weight gray edges must be in every spanning tree. Let's start with another lemma:

Lemma 12.11.14. *[Gray Edge Swap] Let G be a connected weighted graph, and let e be the minimum-weight gray edge of some black-white coloring of of G. Suppose C is a connected spanning subgraph of G, and e is* not *an edge of C. Then there is an edge g of C such that*

- *$\text{weight}(e) < \text{weight}(g)$,*
- *$C - g + e$ is connected, where $C - g + e$ is the graph obtained by removing g and adding e to the set of edges of C.*

Notice that the Gray Edge Lemma 12.11.11 now follows immediately from the Swap Lemma 12.11.14: if C is a spanning tree of G, then $C - g + e$ will also be a spanning tree, and it has smaller weight. So C cannot be minimum-weight unless e is one of its edges.

Proof. (Swap Lemma 12.11.14) Pick a black-white coloring for which e is a min-weight gray edge. Since C is a connected spanning graph, there is a path $\mathbf{p}$ in C between the different colored endpoints of e. Now $\mathbf{p}$ has both a black endpoint and a white one, so it must contain some gray edge $g \neq e$. Since e is a minimum weight gray edge and there are no same-weight edges, we have

$$\text{weight}(e) < \text{weight}(g).$$

Next, notice that $\mathbf{p} + e$ is a cycle. So when g is removed from the cycle, the endpoints of g remain connected by the rest of the cycle. This means $C + e - g$ is still a connected, spanning subgraph, as required. ■

12.12 References

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

Problems for Section 12.2

Practice Problems

Problem 12.1.
The average degree of the vertices in an n-vertex graph is twice the average number

of edges per vertex. Explain why.

Problem 12.2.
Among *connected* simple graphs whose sum of vertex degrees is 20:

(a) What is the largest possible number of vertices?

Briefly explain:

(b) What is the smallest possible number of vertices?

Briefly explain:

Class Problems

Problem 12.3. (a) Prove that in every simple graph, there are an even number of vertices of odd degree.

(b) Conclude that at a party where some people shake hands, the number of people who shake hands an odd number of times is an even number.

(c) Call a sequence of people at the party a *handshake sequence* if each person in the sequence has shaken hands with the next person, if any, in the sequence.

Suppose George was at the party and has shaken hands with an odd number of people. Explain why, starting with George, there must be a handshake sequence ending with a different person who has shaken an odd number of hands.

Exam Problems

Problem 12.4.
A researcher analyzing data on heterosexual sexual behavior in a group of m males and f females found that within the group, the male average number of female partners was 10% larger that the female average number of male partners.

(a) Comment on the following claim. “Since we’re assuming that each encounter involves one man and one woman, the average numbers should be the same, so the males must be exaggerating.”

(b) For what constant c is $m = c \cdot f$?

(c) The data shows that approximately 20% of the females were virgins, while only 5% of the males were. The researcher wonders how excluding virgins from the population would change the averages. If he knew graph theory, the researcher would realize that the nonvirgin male average number of partners will be $x(f/m)$ times the nonvirgin female average number of partners. What is x?

(d) For purposes of further research, it would be helpful to pair each female in the group with a unique male in the group. Explain why this is not possible.

Problems for Section 12.4

Practice Problems

Problem 12.5.
Which of the items below are simple-graph properties preserved under isomorphism?

(a) There is a cycle that includes all the vertices.

(b) The vertices are numbered 1 through 7.

(c) The vertices can be numbered 1 through 7.

(d) There are two degree 8 vertices.

(e) Two edges are of equal length.

(f) No matter which edge is removed, there is a path between any two vertices.

(g) There are two cycles that do not share any vertices.

(h) The vertices are sets.

(i) The graph can be drawn in a way that all the edges have the same length.

(j) No two edges cross.

(k) The OR of two properties that are preserved under isomorphism.

(l) The negation of a property that is preserved under isomorphism.

Class Problems

Problem 12.6.
For each of the following pairs of simple graphs, either define an isomorphism between them, or prove that there is none. (We write ab as shorthand for $\langle a—b \rangle$.)

(a)

$$G_1 \text{ with } V_1 = \{1,2,3,4,5,6\},\ E_1 = \{12,23,34,14,15,35,45\}$$
$$G_2 \text{ with } V_2 = \{1,2,3,4,5,6\},\ E_2 = \{12,23,34,45,51,24,25\}$$

(b)

$$G_3 \text{ with } V_3 = \{1,2,3,4,5,6\},\ E_3 = \{12,23,34,14,45,56,26\}$$
$$G_4 \text{ with } V_4 = \{a,b,c,d,e,f\},\ E_4 = \{ab,bc,cd,de,ae,ef,cf\}$$

Problem 12.7.
List all the isomorphisms between the two graphs given in Figure 12.23. Explain why there are no others.

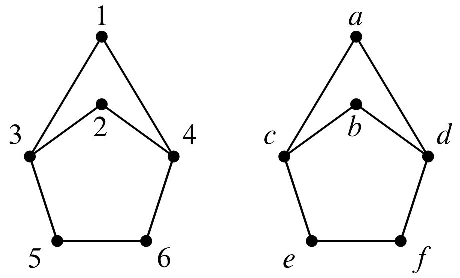


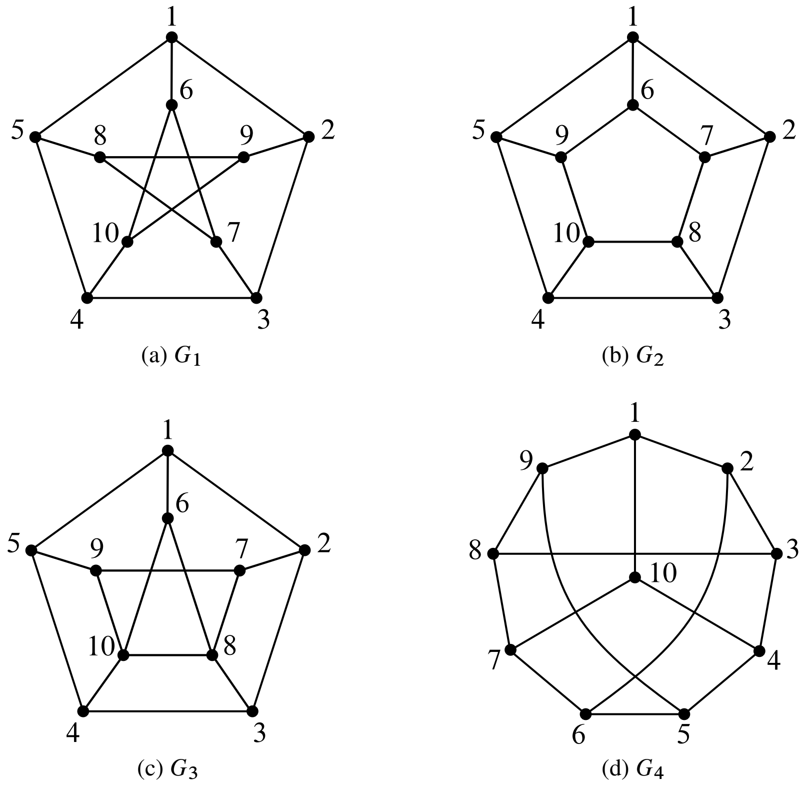
Figure 12.23 Graphs with several isomorphisms

Homework Problems

Problem 12.8.
Determine which among the four graphs pictured in Figure 12.24 are isomorphic. For each pair of isomorphic graphs, describe an isomorphism between them. For each pair of graphs that are not isomorphic, give a property that is preserved under isomorphism such that one graph has the property, but the other does not. For at least one of the properties you choose, *prove* that it is indeed preserved under isomorphism (you only need prove one of them).

Problem 12.9. (a) For any vertex v in a graph, let $\text{N}(v)$ be the set of *neighbors* of v, namely, the vertices adjacent to v:

$$\text{N}(v) ::= \{u \mid \langle u—v \rangle \text{ is an edge of the graph}\}.$$



(a) G_1

(b) G_2

(c) G_3

(d) G_4

Figure 12.24 Which graphs are isomorphic?

Suppose f is an isomorphism from graph G to graph H. Prove that $f(\mathrm{N}(v)) = \mathrm{N}(f(v))$ for each vertex v of G. In other words, if f maps vertex v to vertex $w = f(v)$, then f maps the set of v's neighbors to precisely the set of w's neighbors.

Your proof should follow by simple reasoning using the definitions of isomorphism and neighbors—no pictures or handwaving.

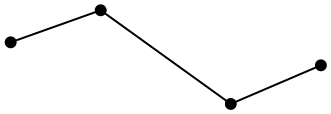
Hint: Prove by a chain of iff's that

$$h \in \mathrm{N}(f(v)) \quad \text{iff} \quad h \in f(\mathrm{N}(v))$$

for every $h \in V_H$. Use the fact that $h = f(u)$ for some $u \in V_G$ (why is this true?).

(b) Conclude that if G and H are isomorphic graphs, then for each $k \in \mathbb{N}$, they have the same number of degree k vertices.

Problem 12.10.
Let's say that a graph has “two ends” if it has exactly two vertices of degree 1 and all its other vertices have degree 2. For example, here is one such graph:



(a) A *line graph* is a graph whose vertices can be listed in a sequence with edges between consecutive vertices only. So the two-ended graph above is also a line graph of length 4.

Prove that the following theorem is false by drawing a counterexample.
False Theorem. *Every two-ended graph is a line graph.*

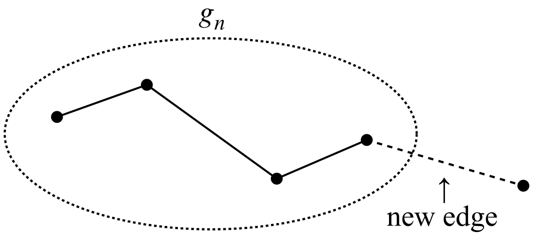
(b) Point out the first erroneous statement in the following bogus proof of the false theorem and describe the error.

Bogus proof. We use induction. The induction hypothesis is that every two-ended graph with n edges is a line graph.

Base case $(n = 1)$**:** The only two-ended graph with a single edge consists of two vertices joined by an edge:

Sure enough, this is a line graph.

Inductive case: We assume that the induction hypothesis holds for some $n \geq 1$ and prove that it holds for $n + 1$. Let G_n be any two-ended graph with n edges. By the induction assumption, G_n is a line graph. Now suppose that we create a two-ended graph G_{n+1} by adding one more edge to G_n. This can be done in only one way: the new edge must join one of the two endpoints of G_n to a new vertex; otherwise, G_{n+1} would not be two-ended.



Clearly, G_{n+1} is also a line graph. Therefore, the induction hypothesis holds for all graphs with $n + 1$ edges, which completes the proof by induction.

■

Problem 12.11.
If G is any simple graph, then a graph isomorphism from G to the same graph G is called a *graph automorphism*[11]. As a simple example, the identity function id : $V(G) \to V(G)$ is always a graph automorphism.

(a) If D is the *Dürer graph* pictured in Figure 12.25, briefly describe a graph automorphism of D that is *not* the identity function.

(b) Define a relation R on $V(G)$ by declaring that $v \; R \; w$ precisely when there exists a graph automorphism f of G with $f(v) = w$. In the special case of the Dürer graph, prove that $1 \; R \; 10$.

Hint: Try to map $1, 2, 3$ to $10, 11, 12$, respectively. Where must the other vertices go?

(c) In the Dürer graph, prove that $\text{NOT}(1 \; R \; 4)$.

Hint: Length 3 cycles.

(d) Prove carefully that for any simple graph G (not necessarily the Dürer graph), the relation R defined above is an equivalence relation.

[11] So-named because "auto" means "self", so an automorphism is a "self-isomorphism."

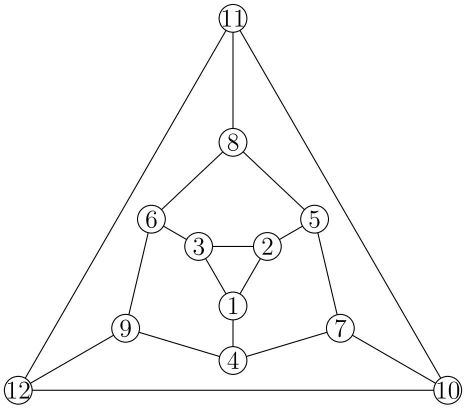


Figure 12.25 The Dürer graph, D.

Hint: If f and g are graph automorphisms, prove that $g \circ f$ is, too.

(e) Because R is an equivalence relation, it partitions the vertices into equivalence classes.[12] What are these equivalence classes for the Dürer graph? How do you know?

Hint: There are only two classes.

Problems for Section 12.5

Practice Problems

Problem 12.12.
Let B be a bipartite graph with vertex sets $L(B)$, $R(B)$. Explain why the sum of the degrees of the vertices in $L(B)$ equals the sum of the degrees of the vertices in $R(B)$.

Class Problems

Problem 12.13.
A certain Institute of Technology has a lot of student clubs; these are loosely overseen by the Student Association. Each eligible club would like to delegate one of its members to appeal to the Dean for funding, but the Dean will not allow a student to

[12] Nodes in the same equivalence class can be thought of, informally, as having the "same role" in the graph, since you can move one to the other through an isomorphism.

be the delegate of more than one club. Fortunately, the Association VP took Math for Computer Science and recognizes a matching problem when she sees one.

(a) Explain how to model the delegate selection problem as a bipartite matching problem. (This is a *modeling problem*; we aren't looking for a description of an algorithm to solve the problem.)

(b) The VP's records show that no student is a member of more than 9 clubs. The VP also knows that to be eligible for support from the Dean's office, a club must have at least 13 members. That's enough for her to guarantee there is a proper delegate selection. Explain. (If only the VP had taken an *Algorithms* class, she could even have found a delegate selection without much effort.)

Problem 12.14.
A simple graph is called *regular* when every vertex has the same degree. Call a graph *balanced* when it is regular and is also a bipartite graph with the same number of left and right vertices.

Prove that if G is a balanced graph, then the edges of G can be partitioned into blocks such that each block is a perfect matching.

For example, if G is a balanced graph with $2k$ vertices each of degree j, then the edges of G can be partitioned into j blocks, where each block consists of k edges, each of which is a perfect matching.

Exam Problems

Problem 12.15.
Marvel is staging 4 test screenings of *Avengers: ∞ War* exclusively for a random selection of MIT students![13] For scheduling purposes, each of the selected students will specify which of the four screenings don't conflict with their schedule–every student is available for at least two out of the four screenings. However, each screening has only 20 available seats, not all of which need to be filled each time. Marvel is thus faced with a difficult scheduling problem: how do they make sure each of the chosen students is able to find a seat at a screening? They've recruited you to help solve this dilemma.

(a) Describe how to model this situation as a matching problem. Be sure to specify what the vertices/edges should be and briefly describe how a matching would determine seat assignments for each student in a screening for which they are available. (This is a *modeling problem*; we aren't looking for a description of an algorithm to

[13] Sadly this isn't actually happening, as far as we know.

solve the problem.)

(b) Suppose 41 students have been selected. Can you guarantee that a matching exists, or are there some situations where not all of the 41 students can be accommodated? Briefly explain.

(c) If instead only 40 students are chosen, prove that there is always a matching.

Hint: Use Hall's Theorem or something similar. Is your graph degree constrained?

Problem 12.16.

Because of the incredible popularity of his class *Math for Computer Science*, TA Mike decides to give up on regular office hours. Instead, he arranges for each student to join some study groups. Each group must choose a representative to talk to the staff, but there is a staff rule that a student can only represent one group. The problem is to find a representative from each group while obeying the staff rule.

(a) Explain how to model the delegate selection problem as a bipartite matching problem. (This is a *modeling problem*; we aren't looking for a description of an algorithm to solve the problem.)

(b) The staff's records show that each student is a member of at most 4 groups, and all the groups have 4 or more members. Is that enough to guarantee there is a proper delegate selection? Explain.

Problem 12.17.

Let $\widehat{R}$ be the "implies" binary relation on propositional formulas defined by the rule that

$$F \ \widehat{R}\ G \quad \text{iff} \quad [(F \text{ IMPLIES } G) \text{ is a valid formula}]. \tag{12.5}$$

For example, $(P \text{ AND } Q)\ \widehat{R}\ P$, because the formula $(P \text{ AND } Q)$ IMPLIES P is valid. Also, it is not true that $(P \text{ OR } Q)\ \widehat{R}\ P$ since $(P \text{ OR } Q)$ IMPLIES P is not valid.

(a) Let A and B be the sets of formulas listed below. Explain why $\widehat{R}$ is not a weak partial order on the set $A \cup B$.

(b) Fill in the $\widehat{R}$ arrows from A to B.

A	arrows	B
		Q
P XOR Q		
		$\overline{P}$ OR $\overline{Q}$
P AND Q		
		$\overline{P}$ OR $\overline{Q}$ OR ($\overline{P}$ AND $\overline{Q}$)
NOT(P AND Q)		
		P

(c) The diagram in part (b) defines a bipartite graph G with $L(G) = A$, $R(G) = B$ and an edge between F and G iff $F \mathrel{\widehat{R}} G$. Exhibit a subset S of A such that both S and $A - S$ are nonempty, and the set $\text{N}(S)$ of neighbors of S is the same size as S, that is, $|\text{N}(S)| = |S|$.

(d) Let G be an arbitrary, finite, bipartite graph. For any subset $S \subseteq L(G)$, let $\overline{S} ::= L(G) - S$, and likewise for any $M \subseteq R(G)$, let $\overline{M} ::= R(G) - M$. Suppose S is a subset of $L(G)$ such that $|\text{N}(S)| = |S|$, and both S and $\overline{S}$ are nonempty. **Circle the formula** that correctly completes the following statement:

There is a matching from $L(G)$ to $R(G)$ if and only if there is both a matching from S to its neighbors, $\text{N}(S)$, and also a matching from $\overline{S}$ to

$\text{N}(\overline{S})$ $\quad$ $\overline{\text{N}(S)}$ $\quad$ $N^{-1}(\text{N}(S))$ $\quad$ $N^{-1}(\text{N}(\overline{S}))$ $\quad$ $\text{N}(\overline{S}) - \overline{\text{N}(S)}$ $\quad$ $\text{N}(S) - \text{N}(\overline{S})$

Hint: The proof of Hall’s Bottleneck Theorem.

Problem 12.18. **(a)** Show that there is no matching for the bipartite graph G in Figure 12.26 that covers $L(G)$.

(b) The bipartite graph H in Figure 12.27 has an easily verified property that implies it has a matching that covers $L(H)$. What is the property?

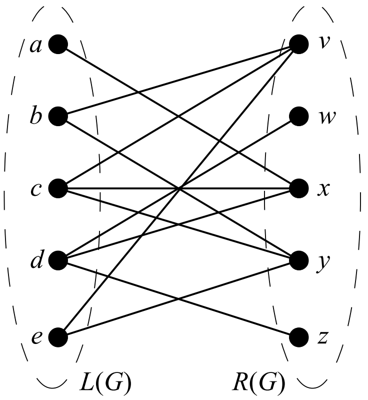


Figure 12.26 Bipartite graph G.

Homework Problems

Problem 12.19.

A *Latin square* is $n \times n$ array whose entries are the number $1, \dots, n$. These entries satisfy two constraints: every row contains all n integers in some order, and also every column contains all n integers in some order. Latin squares come up frequently in the design of scientific experiments for reasons illustrated by a little story in a footnote.[14]

[14] At Guinness brewery in the eary 1900’s, W. S. Gosset (a chemist) and E. S. Beavan (a “maltster”) were trying to improve the barley used to make the brew. The brewery used different varieties of barley according to price and availability, and their agricultural consultants suggested a different fertilizer mix and best planting month for each variety.

Somewhat sceptical about paying high prices for customized fertilizer, Gosset and Beavan planned a season long test of the influence of fertilizer and planting month on barley yields. For as many months as there were varieties of barley, they would plant one sample of each variety using a different one of the fertilizers. So every month, they would have all the barley varieties planted and all the fertilizers used, which would give them a way to judge the overall quality of that planting month. But they also wanted to judge the fertilizers, so they wanted each fertilizer to be used on each variety during the course of the season. Now they had a little mathematical problem, which we can abstract as follows.

Suppose there are n barley varieties and an equal number of recommended fertilizers. Form an $n \times n$ array with a column for each fertilizer and a row for each planting month. We want to fill in the entries of this array with the integers $1,\dots,n$ numbering the barley varieties, so that every row contains all n integers in some order (so every month each variety is planted and each fertilizer is used), and also every column contains all n integers (so each fertilizer is used on all the varieties over the course of the growing season).

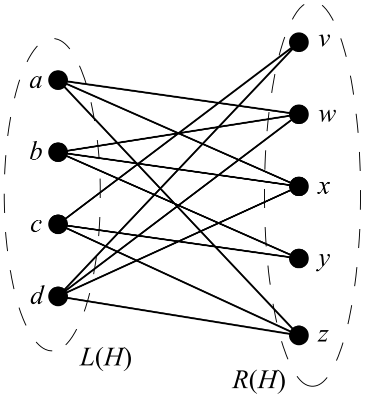


Figure 12.27 Bipartite Graph H.

For example, here is a 4×4 Latin square:

1	2	3	4
3	4	2	1
2	1	4	3
4	3	1	2

(a) Here are three rows of what could be part of a 5×5 Latin square:

2	4	5	3	1
4	1	3	2	5
3	2	1	5	4

Fill in the last two rows to extend this “Latin rectangle” to a complete Latin square.

(b) Show that filling in the next row of an $n \times n$ Latin rectangle is equivalent to finding a matching in some $2n$-vertex bipartite graph.

(c) Prove that a matching must exist in this bipartite graph and, consequently, a Latin rectangle can always be extended to a Latin square.

Problem 12.20.
Take a regular deck of 52 cards. Each card has a suit and a value. The suit is one of four possibilities: heart, diamond, club, spade. The value is one of 13 possibilities, $A, 2, 3, \dots, 10, J, Q, K$. There is exactly one card for each of the 4×13 possible combinations of suit and value.

Ask your friend to lay the cards out into a grid with 4 rows and 13 columns. They can fill the cards in any way they'd like. In this problem you will show that you can always pick out 13 cards, one from each column of the grid, so that you wind up with cards of all 13 possible values.

(a) Explain how to model this trick as a bipartite matching problem between the 13 column vertices and the 13 value vertices. Is the graph necessarily degree-constrained?

(b) Show that any n columns must contain at least n different values and prove that a matching must exist.

Problem 12.21.
Scholars through the ages have identified *twenty* fundamental human virtues: honesty, generosity, loyalty, prudence, completing the weekly course reading-response, etc. At the beginning of the term, every student in Math for Computer Science possessed exactly *eight* of these virtues. Furthermore, every student was unique; that is, no two students possessed exactly the same set of virtues. The Math for Computer Science course staff must select *one* additional virtue to impart to each student by the end of the term. Prove that there is a way to select an additional virtue for each student so that every student is unique at the end of the term as well.

Hint: Look for a matching in an appropriately defined bipartite graph. Be sure to clearly specify your (left and right) vertices and edges.

Problem 12.22.
Suppose n teams play in a round-robin tournament. Each day, each team will play a match with another team. Over a period of $n - 1$ days, every team plays every other team exactly once. There are no ties. Show that for each day we can select a winning team, without selecting the same team twice.[15]

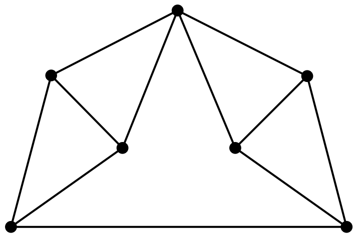
Hint: Define a bipartite graph G with $L(G)$ the set of days and $R(G)$ the set of teams. For any set D of days, there may, or may not, have been a team that lost on all of those days.

[15] Based on 2012 Putnam Exam problem B3.

Problems for Section 12.6

Class Problems

Problem 12.23.
Let G be the graph below.[16] Carefully explain why $\chi(G) = 4$.



Problem 12.24.
A portion of a computer program consists of a sequence of calculations where the results are stored in variables, like this:

$$\begin{array}{rrcl}
 & \text{Inputs:} & & a, b \\
\text{Step 1.} & c & = & a + b \\
2. & d & = & a * c \\
3. & e & = & c + 3 \\
4. & f & = & c - e \\
5. & g & = & a + f \\
6. & h & = & f + 1 \\
 & \text{Outputs:} & & d, g, h
\end{array}$$

A computer can perform such calculations most quickly if the value of each variable is stored in a *register*, a chunk of very fast memory inside the microprocessor. Programming language compilers face the problem of assigning each variable in a program to a register. Computers usually have few registers, however, so they must be used wisely and reused often. This is called the *register allocation* problem.

In the example above, variables a and b must be assigned different registers, because they hold distinct input values. Furthermore, c and d must be assigned different registers; if they used the same one, then the value of c would be overwritten in the second step and we’d get the wrong answer in the third step. On the

[16] From [32], Exercise 13.3.1

other hand, variables b and d may use the same register; after the first step, we no longer need b and can overwrite the register that holds its value. Also, f and h may use the same register; once $f + 1$ is evaluated in the last step, the register holding the value of f can be overwritten.

(a) Recast the register allocation problem as a question about graph coloring. What do the vertices correspond to? Under what conditions should there be an edge between two vertices? Construct the graph corresponding to the example above.

(b) Color your graph using as few colors as you can. Call the computer's registers $R1$, $R2$ etc. Describe the assignment of variables to registers implied by your coloring. How many registers do you need?

(c) Suppose that a variable is assigned a value more than once, as in the code snippet below:

$$\begin{aligned}
&\dots\\
t &= r + s\\
u &= t * 3\\
t &= m - k\\
v &= t + u\\
&\dots
\end{aligned}$$

How might you cope with this complication?

Problem 12.25.
Suppose an n-vertex bipartite graph has exactly k connected components, each of which has two or more vertices. How many ways are there to color it using a given set of two colors?

Homework Problems

Problem 12.26.
6.042 is often taught using recitations. Suppose it happened that 8 recitations were needed, with two or three staff members running each recitation. The assignment of staff to recitation sections, using their secret codenames, is as follows:

- R1: Maverick, Goose, Iceman
- R2: Maverick, Stinger, Viper

- R3: Goose, Merlin
- R4: Slider, Stinger, Cougar
- R5: Slider, Jester, Viper
- R6: Jester, Merlin
- R7: Jester, Stinger
- R8: Goose, Merlin, Viper

Two recitations can not be held in the same 90-minute time slot if some staff member is assigned to both recitations. The problem is to determine the minimum number of time slots required to complete all the recitations.

(a) Recast this problem as a question about coloring the vertices of a particular graph. Draw the graph and explain what the vertices, edges, and colors represent.

(b) Show a coloring of this graph using the fewest possible colors; explain why no fewer colors will work. What schedule of recitations does this imply?

Problem 12.27.
This problem generalizes the result proved Theorem 12.6.3 that any graph with maximum degree at most w is $(w + 1)$-colorable.

A simple graph G is said to have *width* w iff its vertices can be arranged in a sequence such that each vertex is adjacent to at most w vertices that precede it in the sequence. If the degree of every vertex is at most w, then the graph obviously has width at most w—just list the vertices in any order.

(a) Prove that every graph with width at most w is $(w + 1)$-colorable.

(b) Describe a 2-colorable graph with minimum width n.

(c) Prove that the average degree of a graph of width w is at most $2w$.

(d) Describe an example of a graph with 100 vertices, width 3, but *average* degree more than 5.

Problem 12.28.
A sequence of vertices of a graph has *width* w iff each vertex is adjacent to at most w vertices that precede it in the sequence. A simple graph G has width w if there is a width-w sequence of all its vertices.

(a) Explain why the width of a graph must be at least the minimum degree of its vertices.

(b) Prove that if a finite graph has width w, then there is a width-w sequence of all its vertices that ends with a minimum degree vertex.

(c) Describe a simple algorithm to find the minimum width of a graph.

Problem 12.29.
Let G be a simple graph whose vertex degrees are all $\leq k$. Prove by induction on number of vertices that if every connected component of G has a vertex of degree strictly less than k, then G is k-colorable.

Problem 12.30.
A basic example of a simple graph with chromatic number n is the complete graph on n vertices, that is $\chi(K_n) = n$. This implies that any graph with K_n as a subgraph must have chromatic number at least n. It's a common misconception to think that, conversely, graphs with high chromatic number must contain a large complete subgraph. In this problem we exhibit a simple example countering this misconception, namely a graph with chromatic number four that contains no *triangle*—length three cycle—and hence no subgraph isomorphic to K_n for $n \geq 3$. Namely, let G be the 11-vertex graph of Figure 12.28. The reader can verify that G is triangle-free.

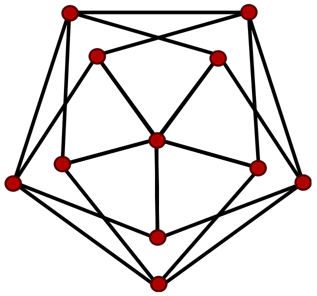


Figure 12.28 Graph G with no triangles and $\chi(G) = 4$.

(a) Show that G is 4-colorable.

(b) Prove that G can't be colored with 3 colors.

Problem 12.31.
This problem will show that 3-coloring a graph is just as difficult as finding a satisfying truth assignment for a propositional formula. The graphs considered will all be taken to have three designated *color-vertices* connected in a triangle to force them to have different colors in any coloring of the graph. The colors assigned to the color-vertices will be called T, F and N.

Suppose f is an n-argument truth function. That is,

$$f : \{T, F\}^n \to \{T, F\}.$$

A graph G is called a *3-color-f-gate* iff G has n designated *input vertices* and a designated *output vertex*, such that

- G can be 3-colored *only* if its input vertices are colored with T's and F's.
- For every sequence $b_1, b_2, \dots, b_n \in \{T, F\}$, there is a 3-coloring of G in which the input vertices $v_1, v_2, \dots, v_n \in V(G)$ have the colors $b_1, b_2, \dots, b_n \in \{T, F\}$.
- In any 3-coloring of G where the input vertices $v_1, v_2, \dots, v_n \in V(G)$ have colors $b_1, b_2, \dots, b_n \in \{T, F\}$, the output vertex has color $f(b_1, b_2, \dots, b_n)$.

For example, a 3-color-NOT-gate consists simply of two adjacent vertices. One vertex is designated to be the input vertex P and the other is designated to be the output vertex. Both vertices have to be constrained so they can only be colored with T's or F's in any proper 3-coloring. This constraint can be imposed by making them adjacent to the color-vertex N, as shown in Figure 12.29.

(a) Verify that the graph in Figure 12.30 is a 3-color-OR-gate. (The dotted lines indicate edges to color-vertex N; these edges constrain the P, Q and P OR Q vertices to be colored T or F in any proper 3-coloring.)

(b) Let E be an n-variable propositional formula, and suppose E defines a truth function $f : \{T, F\}^n \to \{T, F\}$. Explain a simple way to construct a graph that is a 3-color-f-gate.

(c) Explain why an efficient procedure for determining if a graph was 3-colorable would lead to an efficient procedure to solve the satisfiability problem, SAT.

Problem 12.32.
The 3-coloring problem for planar graphs turns out to be no easier than the 3-coloring problem for arbitrary graphs. This claim follows very simply from the

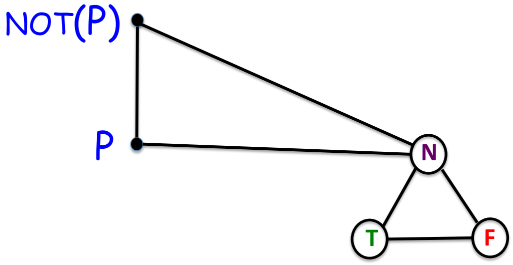


Figure 12.29 A 3-color NOT-gate

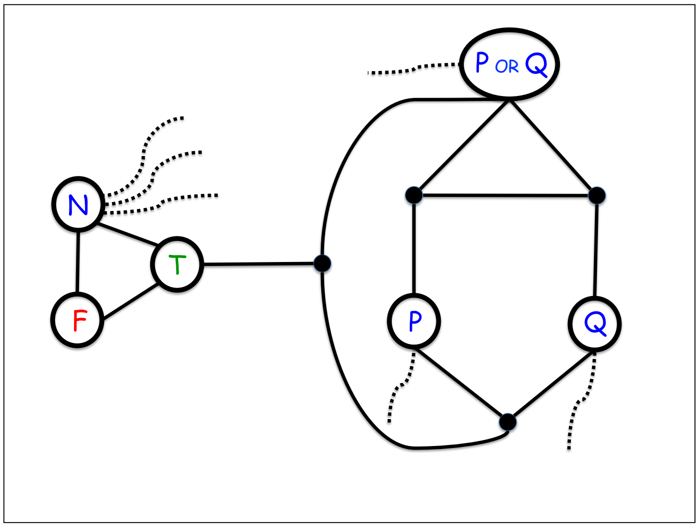


Figure 12.30 A 3-color OR-gate

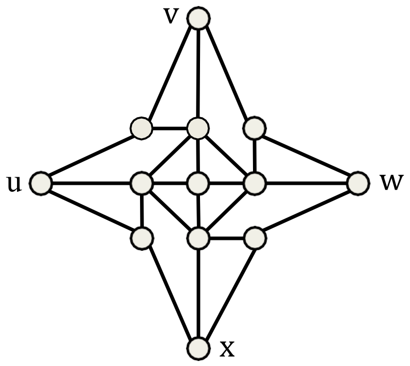


Figure 12.31 A 3-color cross-over gadget.

existence of a "3-color cross-over gadget." Such a gadget is a planar graph whose outer face is a cycle with four designated vertices u, v, w, x occurring in clockwise order such that

(i) Any assignment of colors to vertices u and v can be completed into a 3-coloring of the gadget.

(ii) In every 3-coloring of the gadget, the colors of u and w are the same, and the colors of v and x are the also same.

Figure 12.31 shows such a 3-color cross-over gadget.[17]

So to find a 3-coloring for *any* simple graph, simply draw it in the plane with edges crossing as needed, and then replace each occurrence of an edge crossing by a copy of the gadget as shown in Figure 12.32. This yields a planar graph which has a 3-coloring iff the original graph had one.

(a) Prove that the graph in Figure 12.31 satisfies condition (i) by exhibiting the claimed 3-colorings.

Hint: Only two colorings are needed, one where u and v are the same color and another where they are not the same color.

(b) Prove that the graph in Figure 12.31 satisfies condition (ii).

Hint: The colorings for part (a) are almost completely forced by the coloring of u and v.

[17] The original such gadget and reduction of 3-colorability to planar 3-colorability were due to Larry Stockmeyer [46]. The current simplified gadget is due to M.S. Paterson.

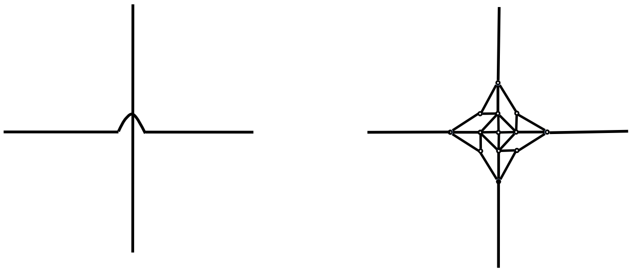


Figure 12.32 Replacing an edge-crossing with a planar gadget.

Exam Problems

Problem 12.33.

False Claim. *Let G be a graph whose vertex degrees are all $\leq k$. If G has a vertex of degree strictly less than k, then G is k-colorable.*

(a) Give a counterexample to the False Claim when $k = 2$.

(b) Underline the exact sentence or part of a sentence that is the first unjustified step in the following bogus proof of the False Claim.

> *Bogus proof.* Proof by induction on the number n of vertices:
>
> The induction hypothesis $P(n)$ is:
>
> > Let G be an n-vertex graph whose vertex degrees are all $\leq k$. If G also has a vertex of degree strictly less than k, then G is k-colorable.
>
> **Base case**: $(n = 1)$ G has one vertex, the degree of which is 0. Since G is 1-colorable, $P(1)$ holds.
>
> **Inductive step**: We may assume $P(n)$. To prove $P(n + 1)$, let G_{n+1} be a graph with $n+1$ vertices whose vertex degrees are all k or less. Also, suppose G_{n+1} has a vertex v of degree strictly less than k. Now we only need to prove that G_{n+1} is k-colorable.
>
> To do this, first remove the vertex v to produce a graph G_n with n vertices. Let u be a vertex that is adjacent to v in G_{n+1}. Removing v reduces the

degree of u by 1. So in G_n, vertex u has degree strictly less than k. Since no edges were added, the vertex degrees of G_n remain $\leq k$. So G_n satisfies the conditions of the induction hypothesis $P(n)$, and so we conclude that G_n is k-colorable.

Now a k-coloring of G_n gives a coloring of all the vertices of G_{n+1}, except for v. Since v has degree less than k, there will be fewer than k colors assigned to the nodes adjacent to v. So among the k possible colors, there will be a color not used to color these adjacent nodes, and this color can be assigned to v to form a k-coloring of G_{n+1}.

■

(c) With a slightly strengthened condition, the preceding proof of the False Claim could be revised into a sound proof of the following Claim:

Claim. *Let G be a graph whose vertex degrees are all $\leq k$.*
If ⟨***statement inserted from below***⟩ *has a vertex of degree strictly less than k, then G is k-colorable.*

Indicate each of the statements below that could be inserted to make the proof correct.

- G is connected and
- G has no vertex of degree zero and
- G does not contain a complete graph on k vertices and
- every connected component of G
- some connected component of G

Problem 12.34.

In the graph shown in Figure 12.33, the vertices connected in the triangle on the left are called *color-vertices*; since they form a triangle, they are forced to have different colors in any coloring of the graph. The colors assigned to the color-vertices will be called **T**, **F** and **N**. The dotted lines indicate edges to the color-vertex **N**.

(a) Explain why for any assignment of *different* truth-colors to P and Q, there is a unique 3-coloring of the graph.

(b) Prove that in any 3-coloring of the whole graph, the vertex labeled P XOR Q is colored with the XOR of the colors of vertices P and Q.

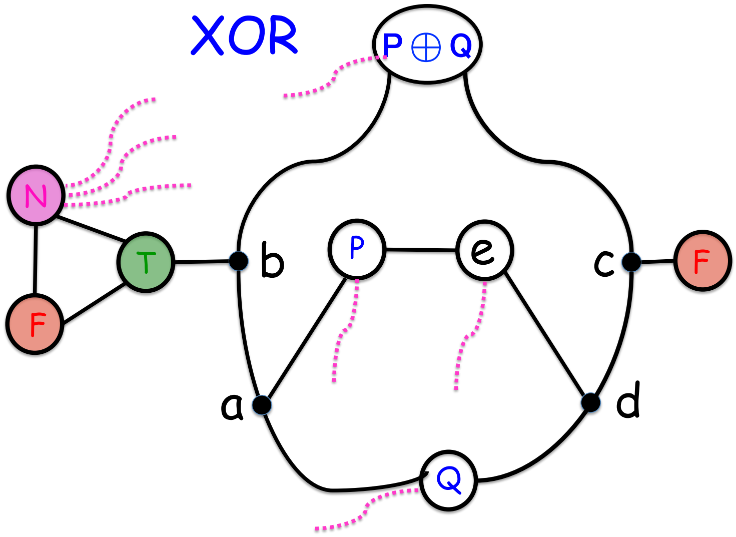


Figure 12.33 A 3-color XOR-gate

Problems for Section 12.7

Practice Problems

Problem 12.35.
Draw a simple graph in which there are at least two distinct paths between two of the graph’s vertices u and v, but neither u nor v is on a cycle. Be sure to label u and v on your drawing.

Hint: There is a five-vertex graph that exhibits this property.

Exam Problems

Problem 12.36.
Since you can go back and forth on an edge in a simple graph, every vertex is on an even length closed walk. So even length closed walks don’t tell you much about even length cycles. The situation with odd-length closed walks is more interesting.

(a) Give an example of a simple graph in which every vertex is on a unique odd-length cycle and a unique even-length cycle.

Hint: Four vertices.

(b) Give an example of a simple graph in which every vertex is on a unique odd-length cycle and no vertex is on an even-length cycle.

(c) Prove that in a digraph, a smallest size odd-length closed walk must be a cycle. Note that there will always be lots of even-length closed walks that are shorter than the smallest odd-length one.

Hint: Let $\mathbf{e}$ be an odd-length closed walk of minimum size, and suppose it begins and ends at vertex a. If it is not a cycle, then it must include a repeated vertex $b \neq a$. That is, $\mathbf{e}$ starts with a walk $\mathbf{f}$ from a to b, followed by a walk $\mathbf{g}$ from b to b, followed by a walk $\mathbf{h}$ from b to a.[18]

Problems for Section 12.10

Class Problems

Problem 12.37.
A simple graph G is *2-removable* iff it contains two vertices $v \neq w$ such that $G - v$ is connected, and $G - w$ is also connected. Prove that every connected graph with at least two vertices is 2-removable.

Hint: Consider a maximum length path.

Problem 12.38.
The n-dimensional *hypercube* H_n is a graph whose vertices are the binary strings of length n. Two vertices are adjacent if and only if they differ in exactly 1 bit. For example, in H_3, vertices `111` and `011` are adjacent because they differ only in the first bit, while vertices `101` and `011` are not adjacent because they differ at both the first and second bits.

(a) Prove that it is impossible to find two spanning trees of H_3 that do not share some edge.

(b) Verify that for any two vertices $x \neq y$ of H_3, there are 3 paths from x to y in H_3, such that, besides x and y, no two of those paths have a vertex in common.

(c) Conclude that the connectivity of H_3 is 3.

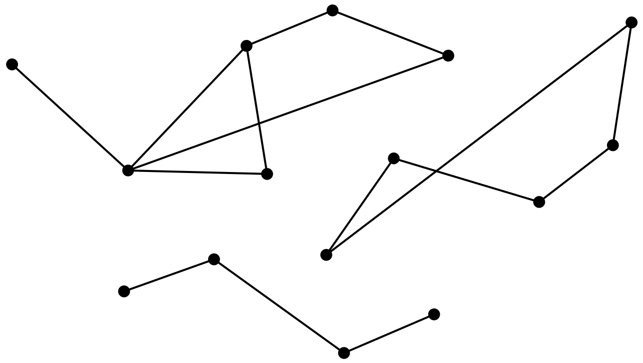
[18] In the notation of the text

$$\mathbf{e} = a\,\mathbf{f}\,\widehat{b}\,\mathbf{g}\,\widehat{b}\,\mathbf{h}\,a.$$

(d) Try extending your reasoning to H_4. (In fact, the connectivity of H_n is n for all $n \geq 1$. A proof appears in the problem solution.)

Problem 12.39.
A set M of vertices of a graph is a *maximal connected* set if every pair of vertices in the set are connected, and any set of vertices properly containing M will contain two vertices that are not connected.

(a) What are the maximal connected subsets of the following (unconnected) graph?



(b) Explain the connection between maximal connected sets and connected components. Prove it.

Problem 12.40. (a) Prove that K_n is $(n-1)$-edge connected for $n > 1$.

Let M_n be a graph defined as follows: begin by taking n graphs with non-overlapping sets of vertices, where each of the n graphs is $(n-1)$-edge connected (they could be disjoint copies of K_n, for example). These will be subgraphs of M_n. Then pick n vertices, one from each subgraph, and add enough edges between pairs of picked vertices that the subgraph of the n picked vertices is also $(n-1)$-edge connected.

(b) Draw a picture of $M_3(\dots M_4)$.

(c) Explain why M_n is $(n-1)$-edge connected.

Problem 12.41.

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

(a) Prove that this Claim is indeed false by providing a counterexample.

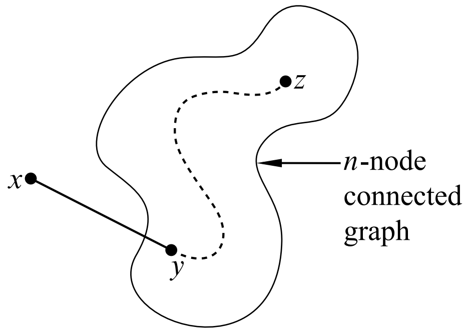
(b) Since the Claim is false, there must be a logical mistake in the following bogus proof. Pinpoint the *first* logical mistake (unjustified step) in the proof.

Bogus proof. We prove the Claim above by induction. Let $P(n)$ be the proposition that if every vertex in an n-vertex graph has positive degree, then the graph is connected.

Base cases: ($n \leq 2$). In a graph with 1 vertex, that vertex cannot have positive degree, so $P(1)$ holds vacuously.

$P(2)$ holds because there is only one graph with two vertices of positive degree, namely, the graph with an edge between the vertices, and this graph is connected.

Inductive step: We must show that $P(n)$ implies $P(n+1)$ for all $n \geq 2$. Consider an n-vertex graph in which every vertex has positive degree. 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:



All that remains is to check that there is a path from x to every other vertex z. Since x has positive degree, there is an edge from x to some other vertex y. Thus, we can obtain a path from x to z by going from x to y and then following the path from y to z. This proves $P(n+1)$.

By the principle of induction, $P(n)$ is true for all $n \geq 0$, which proves the Claim.

■

Homework Problems

Problem 12.42.

An edge is said to *leave* a set of vertices if one end of the edge is in the set and the other end is not.

(a) An n-node graph is said to be *mangled* if there is an edge leaving every set of $\lfloor n/2 \rfloor$ or fewer vertices. Prove the following:

Claim. *Every mangled graph is connected.*

An n-node graph is said to be *tangled* if there is an edge leaving every set of $\lceil n/3 \rceil$ or fewer vertices.

(b) Draw a tangled graph that is not connected.

(c) Find the error in the bogus proof of the following:

False Claim. *Every tangled graph is connected.*

Bogus proof. The proof is by strong induction on the number of vertices in the graph. Let $P(n)$ be the proposition that if an n-node graph is tangled, then it is connected. In the base case, $P(1)$ is true because the graph consisting of a single node is trivially connected.

For the inductive case, assume $n \geq 1$ and $P(1), \ldots, P(n)$ hold. We must prove $P(n+1)$, namely, that if an $(n+1)$-node graph is tangled, then it is connected.

So let G be a tangled, $(n+1)$-node graph. Choose $\lceil n/3 \rceil$ of the vertices and let G_1 be the tangled subgraph of G with these vertices and G_2 be the tangled subgraph with the rest of the vertices. Note that since $n \geq 1$, the graph G has a least two vertices, and so both G_1 and G_2 contain at least one vertex. Since G_1 and G_2 are tangled, we may assume by strong induction that both are connected. Also, since G is tangled, there is an edge leaving the vertices of G_1 which necessarily connects to a vertex of G_2. This means there is a path between any two vertices of G: a path within one subgraph if both vertices are in the same subgraph, and a path traversing the connecting edge if the vertices are in separate subgraphs. Therefore, the entire graph G is connected. This completes the proof of the inductive case, and the Claim follows by strong induction. ■

Problem 12.43.

In the cycle C_{2n} of length $2n$, we'll call two vertices *opposite* if they are on opposite sides of the cycle, that is that are distance n apart in C_{2n}. Let G be the graph formed from C_{2n} by adding an edge, which we'll call a *crossing edge*, between each pair of opposite vertices. So G has n crossing edges.

(a) Give a simple description of the shortest path between any two vertices of G.

Hint: Argue that a shortest path between two vertices in G uses at most one crossing edge.

(b) What is the *diameter* of G, that is, the largest distance between two vertices?

(c) Prove that the graph is not 4-connected.

(d) Prove that the graph is 3-connected.

Problem 12.44.

An *Euler tour* of a graph is a closed walk that includes every edge exactly once.[19] In this problem we work out a proof of:

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

(a) Show that if a graph has an Euler tour, then the degree of each of its vertices is even.

In the remaining parts, we'll work out the converse: if the degree of every vertex of a connected finite graph is even, then it has an Euler tour. To do this, let's define an Euler *walk* to be a walk that includes each edge *at most* once.

(b) Suppose that an Euler walk in a connected graph does not include every edge. Explain why there must be an unincluded edge that is incident to a vertex on the walk.

In the remaining parts, let **w** be the *longest* Euler walk in some finite, connected graph.

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

Hint: part (b)

(d) Explain why all the edges incident to the end of **w** must already be in **w**.

(e) Show that if the end of **w** was not equal to the start of **w**, then the degree of the end would be odd.

Hint: part (d)

(f) Conclude that if every vertex of a finite, connected graph has even degree, then it has an Euler tour.

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

Exam Problems

Problem 12.45.
We apply the following operation to a **simple graph** G: pick two vertices $u \neq v$ such that either

1. there is an edge of G between u and v, and there is also a path from u to v which does *not* include this edge; in this case, delete the edge $\langle u—v \rangle$.

2. there is no path from u to v; in this case, add the edge $\langle u—v \rangle$.

Keep repeating these operations until it is no longer possible to find two vertices $u \neq v$ to which an operation applies.

Assume the vertices of G are the integers $1, 2, \ldots, n$ for some $n \geq 2$. This procedure can be modelled as a state machine whose states are all possible simple graphs with vertices $1, 2, \ldots, n$. G is the start state, and the final states are the graphs on which no operation is possible.

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

$$\{\{1, 2\}, \{3, 4\}\}$$

How many possible final states are reachable from start state G? 1in

(b) On the line next to each of the derived state variables below, indicate the *strongest* property from the list below that the variable is guaranteed to satisfy, no matter what the starting graph G is. The properties are:

constant *increasing* *decreasing*
nonincreasing *nondecreasing* *none of these*

For any state, let e be the number of edges in it, and let c be the number of **connected components** it has. Since e may increase or decrease in a transition, it does not have any of the first four properties. The derived variables are:

0) e *none of these*

i) c

ii) $c + e$

iii) $2c + e$

iv) $c + \frac{e}{e+1}$

(c) Explain why, starting from any state G, the procedure terminates. If your explanation depends on answers you gave to part (b), you must justify those answers.

(d) Prove that any final state must be an **unordered tree** on the set of vertices, that is, a spanning tree.

Problem 12.46.
If a simple graph has e edges, v vertices, and k connected components, then it has at least $e - v + k$ cycles.

Prove this by induction on the number of edges e.

Problems for Section 12.11

Practice Problems

Problem 12.47. (a) Prove that the average degree of a tree is less than 2.

(b) Suppose every vertex in a graph has degree at least k. Explain why the graph has a path of length k.

Hint: Consider a longest path.

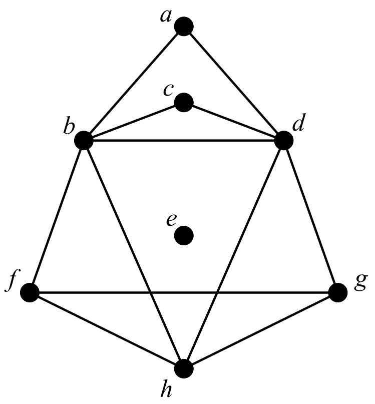


Figure 12.34 The graph G.

Problem 12.48. (a) How many spanning trees are there for the graph G in Figure 12.34?

(b) For $G - e$, the graph G with vertex e deleted, describe two spanning trees that have no edges in common.

(c) For $G - e$ with edge $\langle a\text{—}d \rangle$ deleted, explain why there cannot be two edge-disjoint spanning trees.

Hint: : Count vertices and edges.

Problem 12.49.

Let H_3 be the graph shown in Figure 12.35. Explain why it is impossible to find two spanning trees of H_3 that have no edges in common.

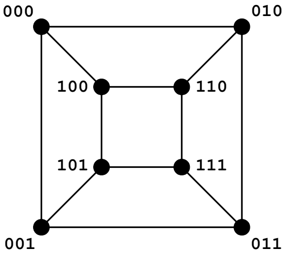


Figure 12.35 H_3 .

Problem 12.50.

The *diameter* of a connected graph is the largest distance between any two vertices.

(a) What is the largest possible diameter in any connected graph with n vertices? Describe a graph with this maximum diameter.

(b) What is the smallest possible diameter of an n-vertex tree for $n > 2$? Describe an n-vertex tree with this minimum diameter.

Problem 12.51.

(a) Indicate all the properties below that are preserved under graph isomorphism.

- There is a cycle that includes all the vertices.
- Two edges are of equal length.

- The graph remains connected if any two edges are removed.
- There exists an edge that is an edge of every spanning tree.
- The negation of a property that is preserved under isomorphism.

(b) For the following statements about **finite trees**, indicate whether they are **true** or **false**, and *provide counterexamples* for those that are **false**.

- Any connected subgraph is a tree. **true** **false**
- Adding an edge between two nonadjacent vertices creates a cycle. **true** **false**
- The number of vertices is one less than twice the number of leaves. **true** **false**
- The number of vertices is one less than the number of edges. **true** **false**
- For every finite graph (not necessarily a tree), there is one (a finite tree) that spans it. **true** **false**

Problem 12.52.

Indicate **True** or **False** for the following statements about finite **simple graphs** G.

(a) G has a spanning tree. **True** **False**

(b) $|V(G)| = O(|E(G)|)$ for connected G. **True** **False**

(c) $|E(G)| = O(|V(G)|)$. **True** **False**

(d) $\chi(G) \le \max\{\deg(v) \mid v \in V(G)\}$.[20] **True** **False**

(e) $|V(G)| = O(\chi(G))$. **True** **False**

Class Problems

Problem 12.53.
Procedure *Mark* starts with a connected, simple graph with all edges unmarked and then marks some edges. At any point in the procedure a path that includes only marked edges is called a *fully marked* path, and an edge that has no fully marked path between its endpoints is called *eligible*.

[20] $\chi(G)$ is the chromatic number of G.

Procedure *Mark* simply keeps marking eligible edges, and terminates when there are none.

Prove that *Mark* terminates, and that when it does, the set of marked edges forms a spanning tree of the original graph.

Problem 12.54.

A procedure for connecting up a (possibly disconnected) simple graph and creating a spanning tree can be modelled as a state machine whose states are finite simple graphs. A state is *final* when no further transitions are possible. The transitions are determined by the following rules:

Procedure create-spanning-tree

1. If there is an edge $\langle u\text{—}v\rangle$ on a cycle, then delete $\langle u\text{—}v\rangle$.
2. If vertices u and v are not connected, then add the edge $\langle u\text{—}v\rangle$.

(a) Draw all the possible final states reachable starting with the graph with vertices $\{1, 2, 3, 4\}$ and edges

$$\{\langle 1\text{—}2\rangle, \langle 3\text{—}4\rangle\}.$$

(b) Prove that if the machine reaches a final state, then the final state will be a tree on the vertices of the graph on which it started.

(c) For any graph G', let e be the number of edges in G', c be the number of connected components it has, and s be the number of cycles. For each of the quantities below, indicate the *strongest* of the properties that it is guaranteed to satisfy, no matter what the starting graph is.

The choices for properties are: *constant*, *strictly increasing*, *strictly decreasing*, *weakly increasing*, *weakly decreasing*, *none of these*.

(i) e

(ii) c

(iii) s

(iv) $e - s$

(v) $c + e$

(vi) $3c + 2e$

(vii) $c + s$

(d) Prove that one of the quantities from part (c) strictly decreases at each transition. Conclude that for every starting state, the machine will reach a final state.

Problem 12.55.
Let G be a weighted graph and suppose there is a unique edge $e \in E(G)$ with smallest weight, that is, $w(e) < w(f)$ for all edges $f \in E(G) - \{e\}$. Prove that any minimum weight spanning tree (MST) of G must include e.

Problem 12.56.
Let G be the 4×4 grid with vertical and horizontal edges between neighboring vertices and edge weights as shown in Figure 12.36.

In this problem you will practice some of the ways to build minimum-weight spanning trees. For each part, list the edge weights in the order in which the edges with those weights were chosen by the given rules.

(a) Construct a minimum weight spanning tree (MST) for G by initially selecting the minimum weight edge, and then successively selecting the minimum weight edge that does not create a cycle with the previously selected edges. Stop when the selected edges form a spanning tree of G. (This is Kruskal's MST algorithm.)

For any step in Kruskal's procedure, describe a black-white coloring of the graph components so that the edge Kruskal chooses is the minimum weight "gray edge" according to Lemma 12.11.10.

(b) Grow an MST for G by starting with the tree consisting of the single vertex u and then successively adding the minimum weight edge with exactly one endpoint in the tree. Stop when the tree spans G. (This is Prim's MST algorithm.)

For any step in Prim's procedure, describe a black-white coloring of the graph components so that the edge Prim chooses is the minimum weight "gray edge"according to Lemma 12.11.10.

(c) The 6.042 "parallel" MST algorithm can grow an MST for G by starting with the upper left corner vertex along with the vertices labelled v and w. Regard each of the three vertices as one-vertex trees. Successively add, for each tree in parallel, the minimum-weight edge among the edges with exactly one endpoint in the tree. Stop working on a tree when there is an edge connecting it to another tree. Continue until there are no more eligible trees—that is, each tree is connected by an edge

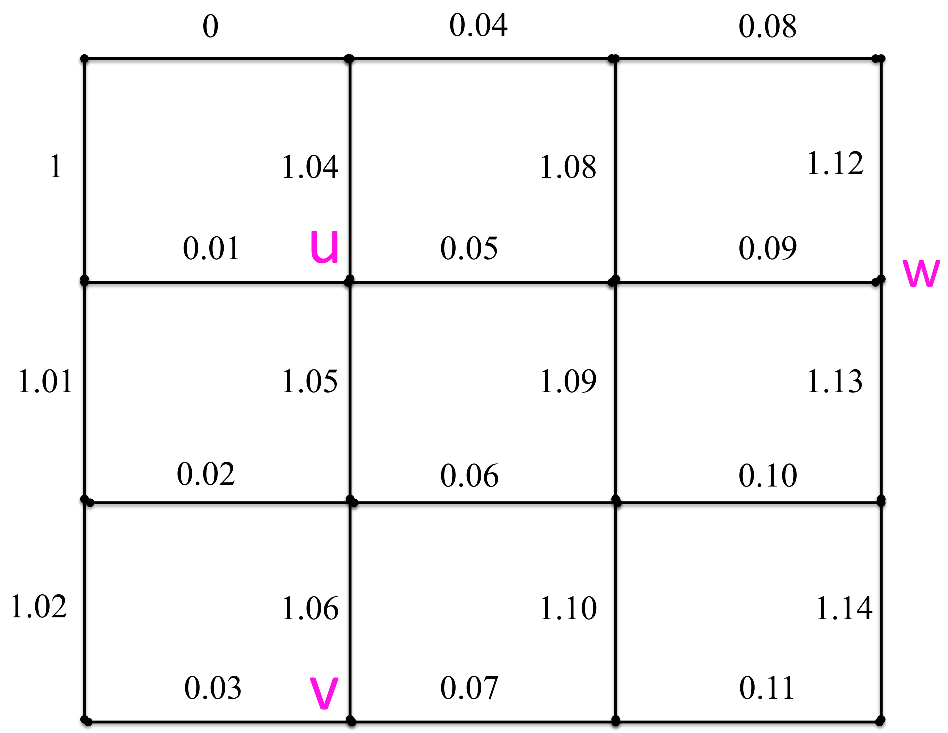


Figure 12.36 The 4x4 array graph G

to another tree—then go back to applying the general gray-edge method until the parallel trees merge to form a spanning tree of G.

(d) Verify that you got the same MST each time as promised by Corollary 12.11.13.

Problem 12.57.
In this problem you will prove:

Theorem. *A graph G is 2-colorable iff it contains no odd length closed walk.*

As usual with “iff” assertions, the proof splits into two proofs: part (a) asks you to prove that the left side of the “iff” implies the right side. The other problem parts prove that the right side implies the left.

(a) Assume the left side and prove the right side. Three to five sentences should suffice.

(b) Now assume the right side. As a first step toward proving the left side, explain why we can focus on a single connected component H within G.

(c) As a second step, explain how to 2-color any tree.

(d) Choose any 2-coloring of a spanning tree T of H. Prove that H is 2-colorable by showing that any edge *not* in T must also connect different-colored vertices.

Problem 12.58.

MIT Information Services & Technology (IS&T) wants to assemble a cluster of n computers using wires and hubs. Each computer must have exactly one wire attached to it, while each hub can have up to five wires attached. There must be a path of wires between every pair of computers in the cluster.

(a) Prove by induction that $\lceil (n-2)/3 \rceil$ hubs are sufficient for IS&T to assemble the cluster of n computers.

Suppose IS&T uses m hubs.

(b) Write a simple formula in terms of m and n that bounds the maximum number of wires that can be attached to the hubs and computers without leaving dangling ends.

(c) Write a simple formula in terms of m and n for a lower bound on the number of wires needed to form the cluster, even using hubs that may have more than five wires attached.

(d) IS&T wants to minimize the number of hubs used to assemble the cluster. Use the previous results to prove that at $\lceil (n-2)/3 \rceil$ hubs are necessary and sufficient to hook up the cluster.

Problem 12.59.

Note that the two parts of this problem do not depend on each other.

(a) Let G be a connected simple graph. Prove that every spanning tree of G must include every cut edge of G.

(b) Suppose a connected, weighted graph G has a unique maximum-weight edge e. Show that if e is in a minimum weight spanning tree of G, then e is a cut edge.

Homework Problems

Problem 12.60.
If a finite simple graph G has exactly $|V(G)| - |E(G)|$ components, then G is a forest. Prove this by induction on the number of vertices.

Problem 12.61.
Let G be a connected simple graph, T be a spanning tree of G, and e be an edge of G.

(a) Prove that if e is *not* on a cycle in G, then e is an edge of T.

(b) Prove that if e *is* on a cycle in G, and e is in T, then there is an edge $f \neq e$ such that $T - e + f$ is also a spanning tree.

(c) Suppose G is edge-weighted, the weight of e is larger than the weights of all the other edges, e is on a cycle in G, and e is an edge of T. Conclude that T is *not* a minimum weight spanning tree of G.

Altogether, we have now shown that a maximum-weight edge is a member of a minimum-weight spanning tree iff it is a cut edge.

Exam Problems

Problem 12.62. **(a)** Let T be a tree and e a new edge between two vertices of T. Explain why $T + e$ must contain a cycle.

(b) Conclude that $T + e$ must have at least three spanning trees.

Problem 12.63.
How many 5-vertex non-isomomorphic trees are there? Explain.

Problem 12.64.
A simple graph G is said to have *width* 1 iff there is a way to list all its vertices so that each vertex is adjacent to at most one vertex that appears earlier in the list. All the graphs mentioned below are assumed to be finite.

(a) Prove that every graph with width one is a forest.

Hint: By induction, removing the last vertex.

(b) Prove that every finite tree has width one. Conclude that a graph is a forest iff it has width one.

Problem 12.65.
Prove by induction that, using a fixed set of $n > 1$ colors, there are exactly

$$n \cdot (n-1)^{m-1}$$

different colorings of any tree with m vertices.

Problem 12.66.
Let G be a connected weighted simple graph and let v be a vertex of G. Suppose $e ::= \langle v—w \rangle$ is an edge of G that is strictly smaller than the weight of every other edge incident to v. Let T be a minimum weight spanning tree of G.

Give a direct proof that e is an edge of T (without appeal to any “gray edge” argument).

Problem 12.67.
For any simple graph G, a “new” edge e is one that connects nonadjacent vertices. (More formally, e is “new” when $e \in V(G)^2 - E(G)$.) let $G + e$ be the graph that results from adding the new edge e to $E(G)$. A real-valued function f on finite simple graphs is

- *strictly edge-increasing* when $f(G + e) > f(G)$,
- *constant* when $f(G) = f(G + e)$
- *weakly edge-increasing* when $f(G + e) \geq f(G)$ and f is not constant,
- *strictly edge-decreasing* when $f(G + e) < f(G)$,
- *weakly edge-decreasing* when $f(G + e) \leq f(G)$ and f is not constant,

for all finite simple graphs G and new edges e.

For example, if $f(G)$ is $|E(G)|$, then $f(G+e) = 1 + f(G) > f(G)$, so this f is strictly increasing. Similarly, if $f(G)$ is $|V(G)|$, then $f(G + e) = f(G)$ since adding an edge between vertices leaves the set of vertices unchanged; this f would be constant.

For each of the following functions $f(G)$, indicate which of the above properties it has, if any.

(i) $\chi(G)$, the chromatic number of G.

(ii) the *connectivity* of G: $\max\{k \in \mathbb{N} \mid G \text{ is } k\text{-connected}\}$

(iii) the maximum path length in G

(iv) the largest *distance*[21] between two vertices of G.

(v) the *girth* of G: the length of the smallest cycle in G

(vi) the number of connected components of G

(vii) the size of the largest complete subgraph of G: $\max\{n \mid K_n \text{ is a subgraph of } G\}$

(viii) the sum of the vertex degrees of G

(ix) the number of spanning trees of G

(x) the number of cut edges in G

[21] The distance between two vertices is the length of the shortest path between them. It is infinite if the vertices are not connected.