Chapter 10

Simple Graphs

Graphs in which edges are *not* directed are called *simple graphs*. They come up in all sorts of applications, including scheduling, optimization, communications, and the design and analysis of algorithms. Two Stanford students even used graph theory to become multibillionaires!

But we'll start with an application designed to get your attention: we are going to make a professional inquiry into sexual behavior. Namely, 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 on average men have 74% more opposite-gender partners than women.

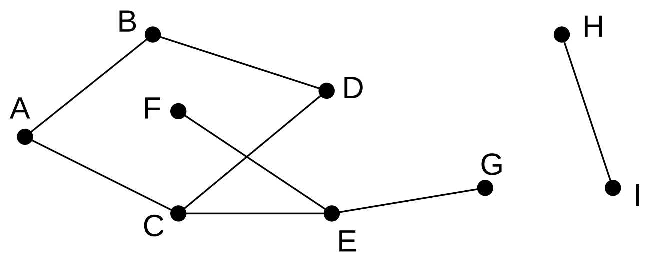
Other studies have found that the disparity is even larger. In particular, ABC News claimed that the average man has 20 partners over his lifetime, and the average woman has 6, for a percentage disparity of 233%. The ABC News study, aired on Primetime Live in 2004, purported to be one of the most scientific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

Yet again, in August, 2007, the N.Y. Times reported on a study by the National Center for Health Statistics of the U.S. government showing that men had seven partners while women had four. Anyway, whose numbers do you think are more accurate, the University of Chicago, ABC News, or the National Center? —don't answer; this is a setup question like "When did you stop beating your wife?" Using a little graph theory, we'll explain why none of these findings can be anywhere near the truth.

10.1 Degrees & Isomorphism

10.1.1 Definition of Simple Graph

Informally, a graph is a bunch of dots with lines connecting some of them. Here is an example:



For many mathematical purposes, we don't really care how the points and lines are laid out —only which points are connected by lines. The definition of *simple graphs* aims to capture just this connection data.

Definition 10.1.1. A *simple graph*, G, consists of a nonempty set, V, called the *vertices* of G, and a collection, E, of two-element subsets of V. The members of E are called the *edges* of G.

The vertices correspond to the dots in the picture, and the edges correspond to the lines. For example, the connection data given in the diagram above can also be given by listing the vertices and edges according to the official definition of simple graph:

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

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

So the definition of simple graphs is the same as for directed graphs, except that instead of a directed edge $v \to w$ which starts at vertex v and ends at vertex w, a simple graph only has an undirected edge, v—w, that connects v and w.

Definition 10.1.2. Two vertices in a simple graph are said to be *adjacent* if they are joined by an edge, and an edge is said to be *incident* to the vertices it joins. The number of edges incident to a vertex is called the *degree* of the vertex; equivalently, the degree of a vertex is equals the number of vertices adjacent to it.

For example, in the simple graph above, A is adjacent to B and B is adjacent to D, and the edge A—C is incident to vertices A and C. Vertex H has degree 1, D has degree 2, and E has degree 3.

Graph Synonyms

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

Some technical consequences of Definition 10.1.1 are worth noting right from the start:

1. Simple graphs do *not* have self-loops ($\{a, a\}$ is not an undirected edge because an undirected edge is defined to be a set of *two* vertices.)

2. There is at most one edge between two vertices of a simple graph.

3. Simple graphs have at least one vertex, though they might not have any edges.

There's no harm in relaxing these conditions, and some authors do, but we don't need self-loops, multiple edges between the same two vertices, or graphs with no vertices, and it's simpler not to have them around.

For the rest of this Chapter we'll only be considering simple graphs, so we'll just call them "graphs" from now on.

10.1.2 Sex 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.[1] We'll put an edge between a male and a female iff they have been sexual partners. This graph is pictured in Figure 10.1 with males on the left and females on the right.

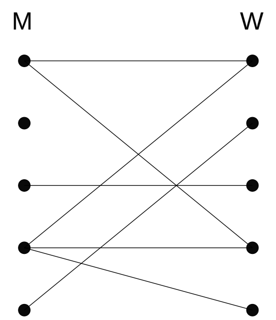


Figure 10.1: The sex partners graph

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

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 is incident to exactly one 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 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 a F* vertex. So we know:

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

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 on average, males have 3.5% more opposite-gender partners than females, and 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 minority students than non-minority students, which is, of course what "minority" means.

10.1.3 Handshaking Lemma

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

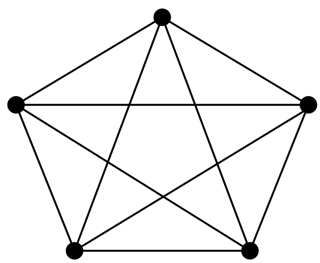
Lemma 10.1.3. *The sum of the degrees of the vertices in a graph equals twice the number of edges.*

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

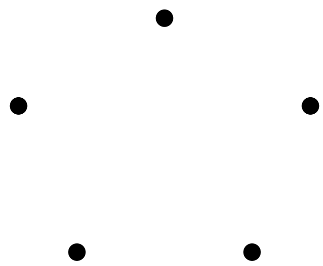
Lemma 10.1.3 is sometimes called the *Handshake Lemma*: if we total up the number of people each person at a party shakes hands with, the total will be twice the number of handshakes that occurred.

10.1.4 Some Common Graphs

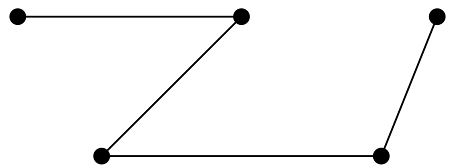
Some graphs come up so frequently that they have names. The *complete graph* on n vertices, also called K_n, has an edge between every two vertices. Here is K_5:



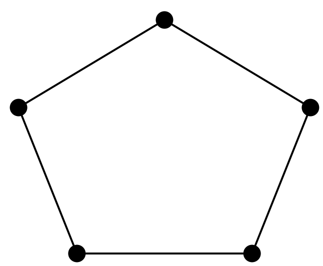
The *empty graph* has no edges at all. Here is the empty graph on 5 vertices:



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

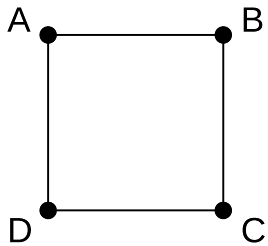


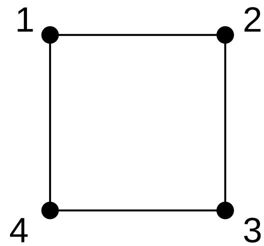
And here is C_5, a *simple cycle* with 5 vertices:



10.1.5 Isomorphism

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





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

Fortunately, we can neatly capture the idea of "looks the same" by adapting Definition 7.2.1 of isomorphism of digraphs to handle simple graphs.

Definition 10.1.4. If G_1 is a graph with vertices, V_1, and edges, E_1, and likewise for G_2, then G_1 is *isomorphic* to G_2 iff there exists a **bijection**, $f : V_1 \to V_2$, such that for every pair of vertices $u, v \in V_1$:

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

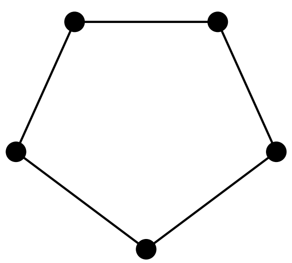
The function f is called an *isomorphism* between G_1 and G_2.

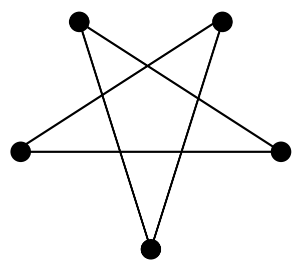
For example, here is an isomorphism between vertices in the two graphs above:

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

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, here are two different ways of drawing C_5:





Isomorphism preserves the connection properties of a graph, abstracting out what the vertices are called, what they are made out of, or where they appear in a drawing of the graph. More precisely, a property of a graph is said to be *preserved under isomorphism* if whenever G has that property, every graph isomorphic to G also has that property. For example, 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. That is, v and $f(v)$ will have the same degree. So if one graph has a vertex of degree 4 and another does not, then they can't be isomorphic.

In fact, they can't be isomorphic if the number of degree 4 vertices in each of the graphs is not the same.

Looking for preserved properties can make it easy to determine that two graphs are not isomorphic, or to actually find an isomorphism between them, if there is one. In practice, it's frequently easy to decide whether two graphs are isomorphic. However, no one has yet found a *general* procedure for determining whether two graphs are isomorphic that is *guaranteed* to run much faster than an exhaustive (and exhausting) search through all possible bijections between their vertices.

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

10.1.6 Problems

Class Problems

Problem 10.1. **(a)** Prove that in every graph, there are an even number of vertices of odd degree.

Hint: The Handshaking Lemma 10.1.3.

(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 two or more different people at the party a *handshake sequence* if, except for the last person, each person in the sequence has shaken hands with the next person 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.

Hint: Just look at the people at the ends of handshake sequences that start with George.

Problem 10.2.
For each of the following pairs of graphs, either define an isomorphism between them, or prove that there is none. (We write ab as shorthand for a—b.)

(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)

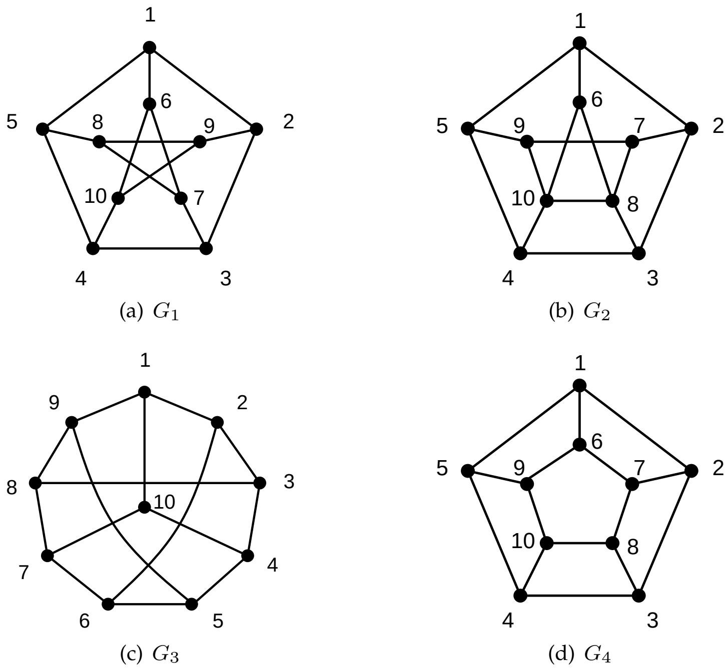
$$G_3 \text{ with } V_3 = \{1,2,3,4,5,6\}\,,\ E_3 = \{12,23,34,14,45,56,26\}$$
$$G_4 \text{ with } V_4 = \{a,b,c,d,e,f\}\,,\ E_4 = \{ab,bc,cd,de,ae,ef,cf\}$$

(c)

$$G_5 \text{ with } V_5 = \{a,b,c,d,e,f,g,h\}\,,\ E_5 = \{ab,bc,cd,ad,ef,fg,gh,he,dh,bf\}$$
$$G_6 \text{ with } V_6 = \{s,t,u,v,w,x,y,z\}\,,\ E_6 = \{st,tu,uv,sv,wx,xy,yz,wz,sw,vz\}$$

Homework Problems

Problem 10.3.
Determine which among the four graphs pictured in the Figures are isomorphic. If two of these graphs are isomorphic, describe an isomorphism between them. If they are not, 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).



(a) G_1 (b) G_2 (c) G_3 (d) G_4

Figure 10.2: Which graphs are isomorphic?

Problem 10.4. **(a)** For any vertex, v, in a graph, let $N(v)$ be the set of *neighbors* of v, namely, the vertices adjacent to v:

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

Suppose f is an isomorphism from graph G to graph H. Prove that $f(N(v)) = N(f(v))$.

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

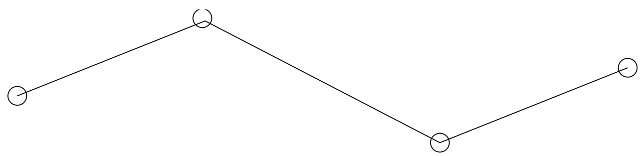
Hint: Prove by a chain of iff's that

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

for every $h \in V_H$. Use the fact that $h = f(u)$ for some $u \in V_G$.

(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 10.5.
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:



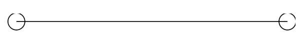
(a) A *line graph* is a graph whose vertices can be listed in a sequence with edges between consecutive vertices only. So the two-ended graph above is also a line graph of length 4.

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 alleged proof of the false theorem. Describe the error as best you can.

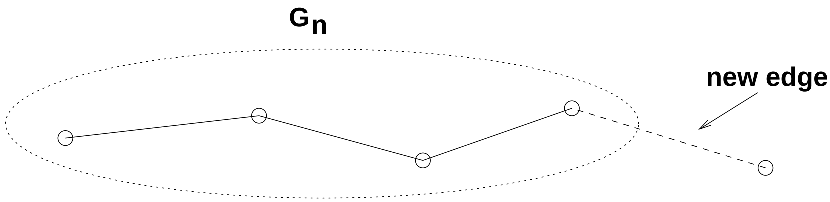
False proof. We use induction. The induction hypothesis is that every two-ended graph with n edges is a path.

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 an endpoint of G_n to a new vertex; otherwise, G_{n+1} would not be two-ended.

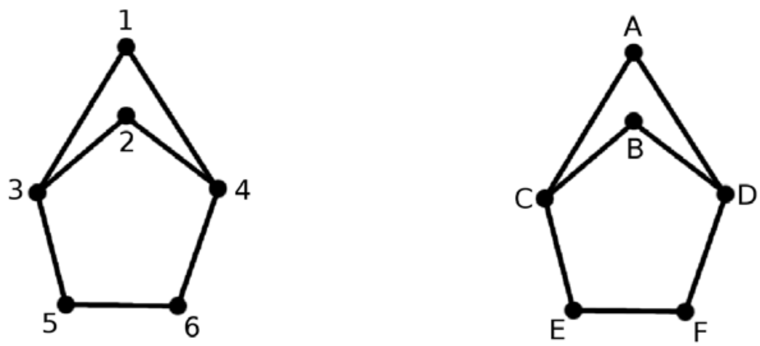


Clearly, G_{n+1} is also a line graph. Therefore, the induction hypothesis holds for all graphs with $n + 1$ edges, which completes the proof by induction.

■

Exam Problems

Problem 10.6.
There are four isomorphisms between these two graphs. List them.



Problem 10.7.
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) Circle all of the assertions below that are implied by the above information on average numbers of partners:

(i) males exaggerate their number of female partners

(ii) $m = (9/10)f$

(iii) $m = (10/11)f$

(iv) $m = (11/10)f$

(v) there cannot be a perfect matching with each male matched to one of his female partners

(vi) there cannot be a perfect matching with each female matched to one of her male partners

(b) 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?

10.2 Connectedness

10.2.1 Paths and Simple Cycles

Paths in simple graphs are esentially the same as paths in digraphs. We just modify the digraph definitions using undirected edges instead of directed ones. For example, the formal definition of a path in a simple graph is a virtually that same as Definition 8.1.1 of paths in digraphs:

Definition 10.2.1. A *path* in a graph, G, is a sequence of $k \geq 0$ vertices

$$v_0, \ldots, v_k$$

such that v_i—v_{i+1} is an edge of G for all i where $0 \leq i < k$. The path is said to *start* at v_0, to *end* at v_k, and the *length* of the path is defined to be k.

An edge, u—v, is *traversed* n times by the path if there are n different values of i such that v_i—$v_{i+1} = u$—v. The path is *simple*[2] iff all the v_i's are different, that is, if $i \neq j$ implies $v_i \neq v_j$.

For example, the graph in Figure 10.3 has a length 6 simple path A,B,C,D,E,F,G. This is the longest simple path in the graph.

As in digraphs, the length of a path is the total number of times it traverses edges, which is *one less* than its length as a sequence of vertices. For example, the length 6 path A,B,C,D,E,F,G is actually a sequence of seven vertices.

[2]Heads up: what we call "paths" are commonly referred to in graph theory texts as "walks," and simple paths are referred to as just "paths". Likewise, what we will call *cycles* and *simple cycles* are commonly called "closed walks" and just "cycles".

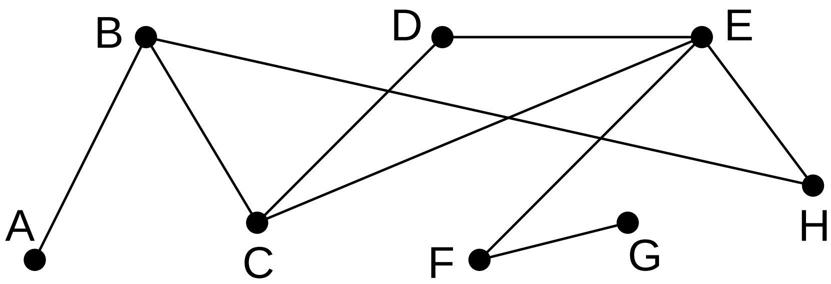


Figure 10.3: *A graph with 3 simple cycles.*

A *cycle* can be described by a path that begins and ends with the same vertex. For example, B,C,D,E,C,B is a cycle in the graph in Figure 10.3. This path suggests that the cycle begins and ends at vertex B, but a cycle isn't intended to have a beginning and end, and can be described by *any* of the paths that go around it. For example, D,E,C,B,C,D describes this same cycle as though it started and ended at D, and D,C,B,C,E,D describes the same cycle as though it started and ended at D but went in the opposite direction. (By convention, a single vertex is a length 0 cycle beginning and ending at the vertex.)

All the paths that describe the same cycle have the same length which is defined to be the *length of the cycle*. (Note that this implies that going around the same cycle twice is considered to be different than going around it once.)

A *simple* cycle is a cycle that doesn't cross or backtrack on itself. For example, the graph in Figure 10.3 has three simple cycles B,H,E,C,B and C,D,E,C and B,C,D,E,H,B. More precisely, a simple cycle is a cycle that can be described by a path of length at least three whose vertices are all different except for the beginning and end vertices. So in contrast to simple *paths*, the length of a simple *cycle* is the *same* as the number of distinct vertices that appear in it.

From now on we'll stop being picky about distinguishing a cycle from a path that describes it, and we'll just refer to the path as a cycle. [3]

Simple cycles are especially important, so we will give a proper definition of them. Namely, we'll define a simple cycle in G to be a *subgraph* of G that looks like a cycle that doesn't cross itself. Formally:

Definition 10.2.2. A *subgraph*, G', of a graph, G, is a graph whose vertices, V', are a subset of the vertices of G and whose edges are a subset of the edges of G.

Notice that since a subgraph is itself a graph, the endpoints of every edge of G'

[3]Technically speaking, we haven't ever defined what a cycle *is*, only how to describe it with paths. But we won't need an abstract definition of cycle, since all that matters about a cycle is which paths describe it.

must be vertices in V'.

Definition 10.2.3. For $n \geq 3$, let C_n be the graph with vertices $1, \ldots, n$ and edges

$$1—2, \;\; 2—3, \;\; \ldots, \;\; (n-1)—n, \;\; n—1.$$

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

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

10.2.2 Connected Components

Definition 10.2.4. Two vertices in a graph are said to be *connected* when there is a path that begins at one and ends at the other. By convention, every vertex is considered to be connected to itself by a path of length zero.

The diagram in Figure 10.4 looks like a picture of three graphs, but is intended to be a picture of *one* graph. This graph consists of three pieces (subgraphs). Each piece by itself is connected, but there are no paths between vertices in different pieces.

Figure 10.4: *One graph with 3 connected components.*

Definition 10.2.5. A graph is said to be *connected* when every pair of vertices are connected.

These connected pieces of a graph are called its *connected components*. A rigorous definition is easy: a connected component is the set of all the vertices connected to some single vertex. So a graph is connected iff it has exactly one connected component. The empty graph on n vertices has n connected components.

10.2.3 How Well Connected?

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

Definition 10.2.6. Two vertices in a graph are *k-edge connected* if they remain connected in every subgraph obtained by deleting $k-1$ edges. A graph with at least two vertices is k-edge connected[4] if every two of its vertices are k-edge connected.

So 1-edge connected is the same as connected for both vertices and graphs. Another way to say that a graph is k-edge connected is that every subgraph obtained from it by deleting at most $k-1$ edges is connected. For example, in the graph in Figure 10.3, vertices B and E are 2-edge connected, G and E are 1-edge connected, and no vertices are 3-edge connected. The graph as a whole is only 1-edge connected. More generally, any simple cycle is 2-edge connected, and the complete graph, K_n, is $(n-1)$-edge connected.

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

10.2.4 Connection by Simple Path

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

Lemma 10.2.7. *If vertex u is connected to vertex v in a graph, then there is a simple path from u to v.*

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

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

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

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

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

yields a strictly shorter path

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

from u to v, contradicting the minimality of the given path. ■

Actually, we proved something stronger:

Corollary 10.2.8. *For any path of length k in a graph, there is a simple path of length* at most *k with the same endpoints.*

10.2.5 The Minimum Number of Edges in a Connected Graph

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

Theorem 10.2.9. *Every graph with v vertices and e edges has at least $v - e$ connected components.*

Of course for Theorem 10.2.9 to be of any use, there must be fewer edges than vertices.

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

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

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

Inductive step: Now we assume that the induction hypothesis holds for every e-edge graph in order to prove that it holds for every $(e+1)$-edge graph, where $e \geq 0$. Consider a graph, G, with $e+1$ edges and k vertices. We want to prove that G has at least $v - (e+1)$ connected components. To do this, remove an arbitrary edge a—b and call the resulting graph G'. By the induction assumption, G' has at least $v - e$ connected components. Now add back the edge a—b to obtain the original graph G. If a and b were in the same connected component of G', then G has the same connected components as G', so G has at least $v - e > v - (e+1)$ components. Otherwise, if a and b were in different connected components of G', then these two components are merged into one in G, but all other components remain unchanged, reducing the number of components by 1. Therefore, G has at least $(v-e)-1 = v-(e+1)$ connected components. So in either case, $P(e+1)$ holds. This completes the Inductive step. The theorem now follows by induction. ■

Corollary 10.2.10. *Every connected graph with v vertices has at least $v - 1$ edges.*

A couple of points about the proof of Theorem 10.2.9 are worth noticing. First, we used induction on the number of edges in the graph. This is very common in proofs involving graphs, and so is induction on the number of vertices. When you're presented with a graph problem, these two approaches should be among the first you consider. The second point is more subtle. Notice that in the inductive step, we took an arbitrary $(n + 1)$-edge graph, threw out an edge so that we could apply the induction assumption, and then put the edge back. You'll see this shrink-down, grow-back process very often in the inductive steps of proofs related to graphs. This might seem like needless effort; why not start with an n-edge graph and add one more to get an $(n + 1)$-edge graph? That would work fine in this case, but opens the door to a nasty logical error called *buildup error*, illustrated in Problems 10.5 and 10.11. Always use shrink-down, grow-back arguments, and you'll never fall into this trap.

10.2.6 Problems

Class Problems

Problem 10.8.
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.

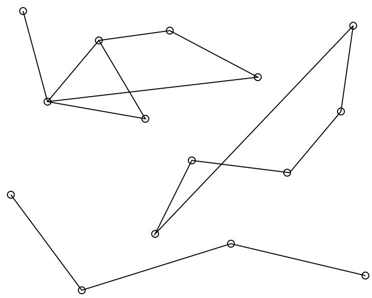
(b) Verify that for any two vertices $x \neq y$ of H_3, there are 3 paths from x to y in H_3, such that, besides x and y, no two of those paths have a vertex in common.

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

(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 10.9.
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 10.10. **(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_4.

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

Problem 10.11.
Definition 10.2.5. A graph is *connected* iff there is a path between every pair of its vertices.

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.

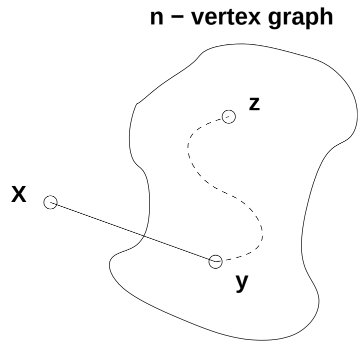
(b) Since the Claim is false, there must be an logical mistake in the following bogus proof. Pinpoint the *first* logical mistake (unjustified step) in the proof.

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.

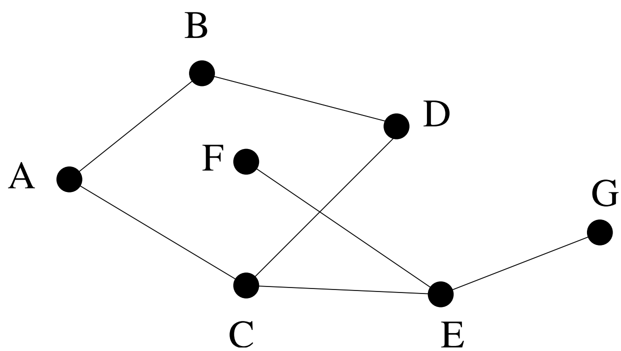
■

Homework Problems

Problem 10.12.
In this problem we'll consider some special cycles in graphs called *Euler circuits*, named after the famous mathematician Leonhard Euler. (Same Euler as for the constant $e \approx 2.718$ —he did a lot of stuff.)

Definition 10.2.11. An Euler circuit of a graph is a cycle which traverses every edge exactly once.

Does the graph in the following figure contain an Euler circuit?



Well, if it did, the edge (E, F) would need to be included. If the path does not start at F then at some point it traverses edge (E, F), and now it is stuck at F since F has no other edges incident to it and an Euler circuit can't traverse (E, F) twice. But then the path could not be a circuit. On the other hand, if the path starts at F, it must then go to E along (E, F), but now it cannot return to F. It again cannot be a circuit. This argument generalizes to show that if a graph has a vertex of degree 1, it cannot contain an Euler circuit.

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

(a) Show that if a graph has an Euler circuit, 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 circuit. To do this, let's define an Euler *path* to be a path that traverses each edge *at most* once.

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

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

(c) Show that if W is a cycle, then it must be an Euler circuit.

Hint: part (b)

(d) Explain why all the edges incident to the end of W must already have been traversed by 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 circuit.

Homework Problems

Problem 10.13.
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.

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 proof of the following

False Claim. *Every tangled graph is connected.*

False 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 10.14.

Let G be the graph formed from C_{2n}, the simple cycle of length $2n$, by connecting every pair of vertices at maximum distance from each other in C_{2n} by an edge in G.

(a) Given two vertices of G find their distance in G.

(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.

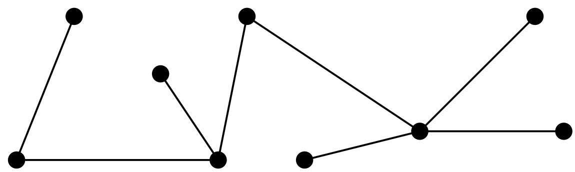
10.3 Trees

Trees are a fundamental data structure in computer science, and there are many kinds, such as rooted, ordered, and binary trees. In this section we focus on the purest kind of tree. Namely, we use the term *tree* to mean a connected graph without simple cycles.

A graph with no simple cycles is called *acyclic*; so trees are acyclic connected graphs.

10.3.1 Tree Properties

Here is an example of a tree:



A vertex of degree at most one is called a *leaf*. In this example, there are 5 leaves. Note that the only case where a tree can have a vertex of degree zero is a graph with a single vertex.

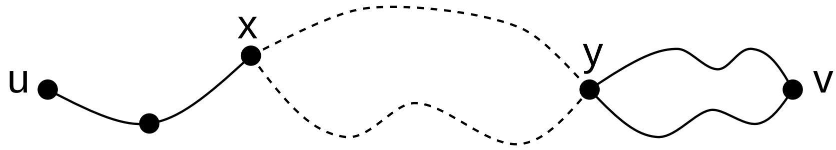
The graph shown above would no longer be a tree if any edge were removed, because it would no longer be connected. The graph would also not remain a tree if any edge were added between two of its vertices, because then it would contain a simple cycle. Furthermore, note that there is a unique path between every pair of vertices. These features of the example tree are actually common to all trees.

Theorem 10.3.1. *Every tree has the following properties:*

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

Proof.

1. A simple cycle in a subgraph is also a simple cycle in the whole graph, so any subgraph of an acyclic graph must also be acyclic. If the subgraph is also connected, then by definition, it is a tree.
2. There is at least one path, and hence one simple path, between every pair of vertices, because the graph is connected. Suppose that there are two different simple paths between vertices u and v. Beginning at u, let x be the first vertex where the paths diverge, and let y be the next vertex they share. Then there are two simple paths from x to y with no common edges, which defines a simple cycle. This is a contradiction, since trees are acyclic. Therefore, there is exactly one simple path between every pair of vertices.



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

4. Suppose that we remove edge u—v. Since the tree contained a unique path between u and v, that path must have been u—v. Therefore, when that edge is removed, no path remains, and so the graph is not connected.

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

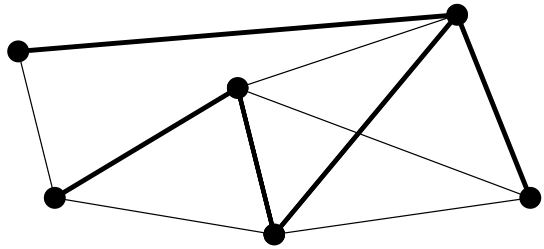
6. We use induction on the number of vertices. For a tree with a single vertex, the claim holds since it has no edges and $0 + 1 = 1$ vertex. Now suppose that the claim holds for all n-vertex trees and consider an $(n+1)$-vertex tree, T. Let v be a leaf of the tree. You can verify that deleting a vertex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

■

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

10.3.2 Spanning Trees

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



Theorem 10.3.2. *Every connected graph contains a spanning tree.*

Proof. Let T be a connected subgraph of G, with the same vertices as G, and with the smallest number of edges possible for such a subgraph. We show that T is acyclic by contradiction. So suppose that T has a cycle with the following edges:

$$v_0 — v_1, v_1 — v_2, \ldots, v_n — v_0$$

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

10.3.3 Problems

Class Problems

Problem 10.15.
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 traverses only marked edges is called a *fully marked* path, and an edge that has no fully marked path between its endpoints is called *eligible*.

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 10.16.

Procedure **create-spanning-tree**

Given a simple graph G, keep applying the following operations to the graph until no operation applies:

1. If an edge u—v of G is on a simple cycle, then delete u—v.
2. If vertices u and v of G are not connected, then add the edge u—v.

Assume the vertices of G are the integers $1, 2, \ldots, n$ for some $n \geq 2$. Procedure **create-spanning-tree** can be modeled as a state machine whose states are all possible simple graphs with vertices $1, 2, \ldots, n$. The start state is G, 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

$$\{1—2, 3—4\}$$

What are the possible final states reachable from start state G? Draw them.

(b) Prove that any final state of must be a tree on the vertices.

(c) For any state, 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 simple cycles. For each of the derived variables below, indicate the *strongest* of the properties that it is guaranteed to satisfy, no matter what the starting graph G is and be prepared to briefly explain your answer.

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

(i) e

(ii) c

(iii) s

(iv) $e - s$

(v) $c + e$

(vi) $3c + 2e$

(vii) $c + s$

(viii) (c, e), partially ordered coordinatewise (the *product* partial order, Ch. 7.4).

(d) Prove that procedure **create-spanning-tree** terminates. (If your proof depends on one of the answers to part (c), you must prove that answer is correct.)

Problem 10.17.

Prove that a graph is a tree iff it has a unique simple path between any two vertices.

Homework Problems

Problem 10.18. **(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 simple path of length k.

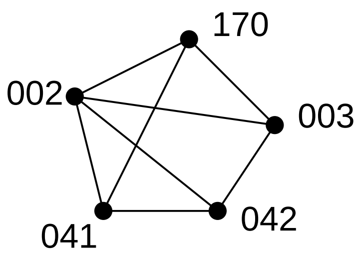
Hint: Consider a longest simple path.

10.4 Coloring Graphs

In section 10.1.2, we used edges to indicate an affinity between two nodes, but having an edge represent a *conflict* between two nodes also turns out to be really useful.

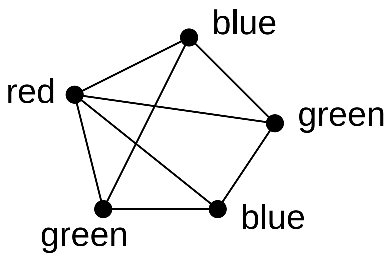
10.5 Modelling Scheduling Conflicts

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 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 exam period would run all year! So, the Schedules Office would also like to keep exam period short. The Schedules Office's problem is easy to describe as a graph. There will be a vertex for each course with a final exam, and two vertices will be adjacent exactly when some student is taking both courses. For example, suppose we need to schedule exams for 6.041, 6.042, 6.002, 6.003 and 6.170. The scheduling graph might look like this:



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). 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 now 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*. For our example graph, three colors suffice:



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

Definition 10.5.1. The minimum value of k for which a graph, G, has a valid coloring is called its *chromatic number*, $\chi(G)$.

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

10.5.1 Degree-bounded Coloring

There are some simple graph properties that give useful upper bounds on colorings. For example, if we have a 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 10.5.2. *A graph with maximum degree at most k is $(k+1)$-colorable.*

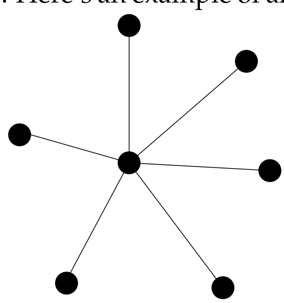
Unfortunately, if you try induction on k, it will lead to disaster. It is not that it is impossible, just that it is extremely painful and would ruin you if you tried it on an exam. Another option, especially with graphs, is to change what you are inducting on. In graphs, some good choices are n, the number of nodes, or e, the number of edges.

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

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

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

Sometimes $k+1$ colors is the best you can do. For example, in the complete graph, K_n, every one of its n vertices is adjacent to all the others, so all n must be assigned different colors. Of course n colors is also enough, so $\chi(K_n) = n$. So K_{k+1} is an example where Theorem 10.5.2 gives the best possible bound. This means that Theorem 10.5.2 also gives the best possible bound for *any* graph with degree bounded by k that has K_{k+1} as a subgraph. But sometimes $k+1$ colors is far from the best that you can do. Here's an example of an n-node star graph for $n = 7$:



In the n-node star graph, the maximum degree is $n-1$, but the star only needs 2 colors!

10.5.2 Why coloring?

One reason coloring problems come all the time is because scheduling conflicts are so common. For example, at Akamai, a new version of software is deployed over each of 20,000 servers every few days. The updates cannot be done at the same time since the servers need to be taken down in order to deploy the software. Also, the servers cannot be handled one at a time, since it would take forever to update them all (each one takes about an hour). Moreover, certain pairs of servers cannot be taken down at the same time since they have common critical functions. This problem was eventually solved by making a 20,000 node conflict graph and coloring it with 8 colors – so only 8 waves of install are needed! Another example comes from the need to assign frequencies to radio stations. If two stations have an overlap in their broadcast area, they can't be given the same frequency. Frequencies are precious and expensive, so you want to minimize the number handed out. This amounts to finding the minimum coloring for a graph whose vertices are the stations and whose edges are between 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, but registers can often 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.

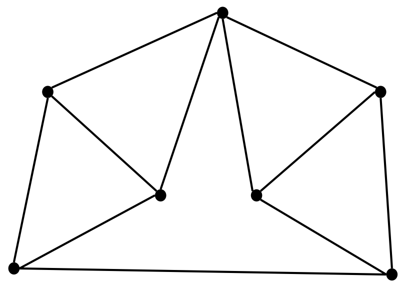
Finally, there's the famous map coloring problem stated in Propostion 1.2.5. The question is how many colors are needed to color a map so that adjacent territories get different colors? This is the same as the number of colors needed to color a graph that can be drawn in the plane without edges crossing. A proof that four colors are enough for the *planar* graphs was acclaimed when it was discovered about thirty years ago. Implicit in that proof was a 4-coloring procedure that takes time proportional to the number of vertices in the graph (countries in the map). On the other hand, it's another of those million dollar prize questions to find an efficient procedure to tell if a planar graph really *needs* four colors or if three will actually do the job. But it's always easy to tell if an *arbitrary* graph is 2-colorable, as we show in Section 10.6. Later in Chapter 12, we'll develop enough planar graph theory to present an easy proof at least that planar graphs are 5-colorable.

10.5.3 Problems

Class Problems

Problem 10.19.
Let G be the graph below[5]. Carefully explain why $\chi(G) = 4$.

[5] From *Discrete Mathematics*, Lovász, Pelikan, and Vesztergombi. Springer, 2003. Exercise 13.3.1



Homework Problems

Problem 10.20.
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 is as follows:

- R1: Eli, Megumi, Rich
- R2: Eli, Stephanie, David
- R3: Megumi, Stav
- R4: Liz, Stephanie, Oscar
- R5: Liz, Tom, David
- R6: Tom, Stav
- R7: Tom, Stephanie
- R8: Megumi, Stav, David

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. What schedule of recitations does this imply?

Problem 10.21.
This problem generalizes the result proved Theorem 10.5.2 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) Describe an example of a graph with 100 vertices, width 3, but *average* degree more than 5. *Hint:* Don't get stuck on this; if you don't see it after five minutes, ask for a hint.

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

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

Exam Problems

Problem 10.22.
Recall that a *coloring* of a graph is an assignment of a color to each vertex such that no two adjacent vertices have the same color. A *k-coloring* is a coloring that uses at most k colors.

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 where the following proof of the False Claim first goes wrong:

False proof. Proof by induction on the number n of vertices:

Induction hypothesis:

$P(n)$::= "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* $\langle$statement inserted from below$\rangle$ *has a vertex of degree strictly less than k, then G is k-colorable.*

Circle each of the statements below that could be inserted to make the Claim true.

- 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

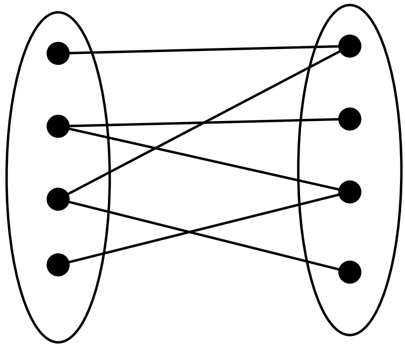
10.6 Bipartite Matchings

10.6.1 Bipartite Graphs

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

Definition 10.6.1. A *bipartite graph* is a graph together with a partition of its vertices into two sets, L and R, such that every edge is incident to a vertex in L and to a vertex in R.

So every bipartite graph looks something like this:



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

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

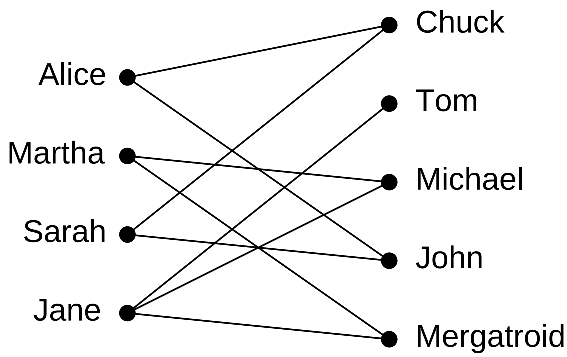
The following Lemma gives another useful characterization of bipartite graphs.

Theorem 10.6.2. *A graph is bipartite iff it has no odd-length cycle.*

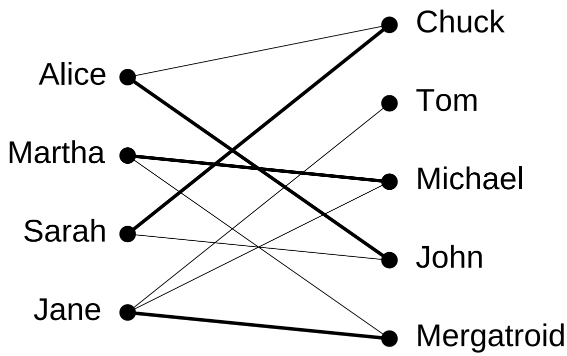
The proof of Theorem 10.6.2 is left to Problem 10.26.

10.6.2 Bipartite Matchings

The *bipartite matching* problem resembles the stable Marriage Problem in that it concerns a set of girls and a set of at least as many boys. There are no preference lists, but each girl does have some boys she likes and others she does not like. In the bipartite matching problem, we ask whether every girl can be paired up with a boy that she likes. Any particular matching problem can be specified by a bipartite graph with a vertex for each girl, a vertex for each boy, and an edge between a boy and a girl iff the girl likes the boy. For example, we might obtain the following graph:



Now a *matching* will mean a way of assigning every girl to a boy so that different girls are assigned to different boys, and a girl is always assigned to a boy she likes. For example, here is one possible matching for the girls:



Hall's Matching Theorem states necessary and sufficient conditions for the existence of a matching in a bipartite graph. It turns out to be a remarkably useful mathematical tool.

10.6.3 The Matching Condition

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

Every subset of girls likes at least as large a set of boys.

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

Theorem 10.6.3. *A matching for a set of girls G with a set of boys B can be found if and only if the matching condition holds.*

Proof. First, let's suppose that a matching exists and show that the matching condition holds. Consider an arbitrary subset of girls. Each girl likes at least the boy she is matched with. Therefore, every subset of girls likes at least as large a set of boys. 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 $|G|$, the number of girls.

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

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

Case 1: Every proper subset of girls likes a *strictly larger* set of boys. In this case, we have some latitude: we pair an arbitrary girl with a boy she likes and send them both away. The matching condition still holds for the remaining boys and girls, so we can match the rest of the girls by induction.

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

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

We sent away $|X|$ girls from the set on the left (leaving X') and sent away an equal number of boys from the set on the right (leaving Y'). Therefore, it must be that $|X'| \leq |Y'|$ as claimed.

So there is in any case a matching for the girls, which completes the proof of the Inductive step. The theorem follows by induction. ■

The proof of this theorem 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, the problem is essentially solved from a computational perspective.

10.6.4 A Formal Statement

Let's restate Hall's Theorem in abstract terms so that you'll not always be condemned to saying, "Now this group of little girls likes at least as many little boys..."

A *matching* in a graph, G, is a set of edges such that no two edges in the set share a vertex. A matching is said to *cover* a set, L, of vertices iff each vertex in L has an edge of the matching incident to it. In any graph, the set $N(S)$, of *neighbors*[6] of some set, S, of vertices is the set of all vertices adjacent to some vertex in S. That is,

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

S is called a *bottleneck* if

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

Theorem 10.6.4 (Hall's Theorem)**.** *Let G be a bipartite graph with vertex partition L, R. There is matching in G that covers L iff no subset of L is a bottleneck.*

An Easy Matching Condition

The bipartite matching condition requires that *every* subset of girls 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 a match and is very easy to check. Namely, call a bipartite graph *degree-constrained* if vertex degrees on the left are at least as large as those on the right. More precisely,

Definition 10.6.5. A bipartite graph G with vertex partition L, R is *degree-constrained* if $\deg(l) \geq \deg(r)$ for every $l \in L$ and $r \in R$.

Now we can always find a matching in a degree-constrained bipartite graph.

Lemma 10.6.6. *Every degree-constrained bipartite graph satisifies the matching condition.*

Proof. Let S be any set of vertices in L. The number of edges incident to vertices in S is exactly the sum of the degrees of the vertices in S. Each of these edges is incident to a vertex in $N(S)$ by definition of $N(S)$. So the sum of the degrees of the vertices in $N(S)$ is at least as large as the sum for S. But since the degree of every vertex in $N(S)$ is at most as large as the degree of every vertex in S, there would have to be at least as many terms in the sum for $N(S)$ as in the sum for S. So there have to be at least as many vertices in $N(S)$ as in S, proving that S is not a bottleneck. So there are no bottlenecks, proving that the degree-constrained graph satisifies the matching condition. ■

[6] An equivalent definition of $N(S)$ uses relational notation: $N(S)$ is simply the image, SR, of S under the adjacency relation, R, on vertices of the graph.

Of course being degree-constrained is a very strong property, and lots of graphs that aren't degree-constrained have matchings. But we'll see examples of degree-constrained graphs come up naturally in some later applications.

10.6.5 Problems

Class Problems

Problem 10.23.
MIT has a lot of student clubs loosely overseen by the MIT 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 be the delegate of more than one club. Fortunately, the Association VP took 6.042 and recognizes a matching problem when she sees one.

(a) Explain how to model the delegate selection problem as a bipartite matching 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 6.046, *Algorithms*, she could even have found a delegate selection without much effort.)

Problem 10.24.
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[7]

[7] 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,…,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).

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.

Exam Problems

Problem 10.25.
Overworked and over-caffeinated, the TAs decide to oust Albert and teach their own recitations. They will run a recitation session at 4 different times in the same room. There are exactly 20 chairs to which a student can be assigned in each recitation. Each student has provided the TAs with a list of the recitation sessions her schedule allows and no student's schedule conflicts with all 4 sessions. The TAs must assign each student to a chair during recitation at a time she can attend, if such an assignment is possible.

(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 recitation that does not conflict with his schedule. (This is a *modeling problem*; we aren't looking for a description of an algorithm to solve the problem.)

(b) Suppose there are 65 students. Given the information provided above, is a matching guaranteed? Briefly explain.

Homework Problems

Problem 10.26.
In this problem you will prove:

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

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 10.27.
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 10.28.
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 6.042 possessed exactly *eight* of these virtues. Furthermore, every student was unique; that is, no two students possessed exactly the same set of virtues. The 6.042 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.

Suggestion: Use Hall's theorem. Try various interpretations for the vertices on the left and right sides of your bipartite graph.

6.042J / 18.062J Mathematics for Computer Science
Spring 2010