Part II

Structures

~~Mathematical Data Types~~

— INSERT 5A goes here —

INSERT 5A

Now that we understand how to construct a good proof, we can turn our attention to the structures that are most useful in modelling problems that arise in computer science. The most important structure is a graph (aka network). Graphs provide an excellent mechanism for modelling associations between pairs of objects (e.g., two exams that cannot be given at the same time, two people that like each other, or two subroutines that can be run independently). In Chapter 5, we study graphs that represent symmetric relationships, and in Chapter 6, we consider graphs where the relationship is one-way (e.g., a situation where you can go from x to y but not necessarily vice-

Versa.

In Chapter 7, we consider the more general notion of a relation and we examine important classes of relations such as ~~equivalence rel~~ partially ordered sets. Partially ordered sets arise frequently in scheduling problems.

We conclude in Chapter 8 with a study of state machines. State machines can be used to model a variety of ~~a~~processes and ~~procedures~~ ~~make~~ are a fundamental ~~the~~ tool ~~it is~~ in proving that an algorithm ~~or the correctness~~ and terminates ~~properties~~ and that it produces the correct output.

Chapter 5

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.

INSERT 5C

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

one needs to be taken before the other.

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

---

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

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~~ Americans over several years to try to get an answer to this question. Their study, published in 1994, and entitled *The Social Organization of Sexuality* found that on average men have 74% more opposite-gender partners than women.

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

~~EDITING NOTE:~~ The promotion for the study is even better:

> A ground breaking ABC News "Primetime Live" survey finds a range of eye-popping sexual activities, fantasies and attitudes in this country,

confirming some conventional wisdom, exploding some myths – and venturing where few scientific surveys have gone before."

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

Yet again, in August, 2007, the N.Y. Times reported on a study by the National Center for Health Statistics of the U.S. government showing that men had seven partners while women had four. Anyway, whose numbers do you think are more accurate, the University of Chicago, ABC News, or the National Center? —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.

9.1 Degrees & Isomorphism

9.1.1 Definition of Simple Graph

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

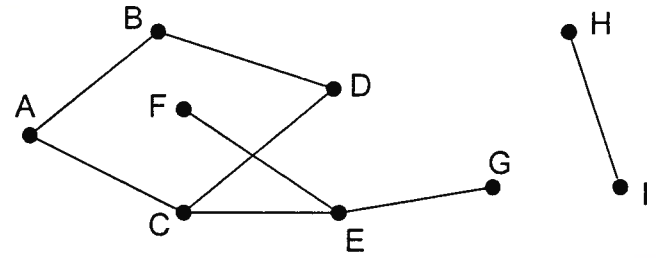


Figure AA: An example of a graph with 9 nodes and 8 edges.

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 graph* aims to capture just this connection data.

Definition 9.1.1. 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 9.1.2. 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 9.1.1 are worth noting right from the start:

1. Simple graphs do *not* have self-loops (i.e., it 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.

deg(E) = 3.

It is possible for a vertex to have degree 0, in which case it is not adjacent to any other vertex. A simple graph does not need to have any edges at all (in which case, the degree of every vertex is zero and $|E| = 0$)[2], but it does need to have at least one vertex (i.e., $|V| \geq 1$).

Note that simple graphs do not have any self-loops (i.e., an edge of the form $\{a, a\}$ since edges are defined to be sets).

An edge is defined to be a set of two vertices. In addition, there is at most one edge between any pair of vertices in a simple graph. That is because E is a set.

Lastly, simple graphs do not contain (and most importantly) directed edges (i.e., edges of the form (a, b) instead of $\{a, b\}$).

In other words, a simple graph does not contain multiple edges or multiple directed edges.

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

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.

9.1.2 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 9.1 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

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

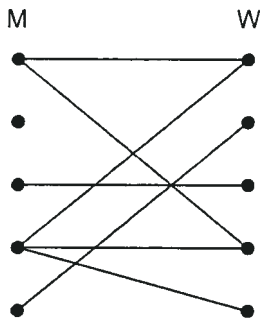


Figure 9.1: The sex partners graph

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, $|M| \cdot |F|$:

$$\left(\frac{\sum_{x \in M} \deg(x)}{|M|}\right) \cdot \frac{1}{|F|} = \left(\frac{\sum_{y \in F} \deg(y)}{|F|}\right) \cdot \frac{1}{|M|} \qquad \text{(Eqn 5A)}$$

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

Notice that

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

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

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

is the average degree of a node in F, which is the average number of opposite-gender partners for a female in America. Hence, Equation 5A implies that on average, an American male has $|F|/|M|$ times as many opposite-gender partners as the average American female.

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.

9.1.3 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:

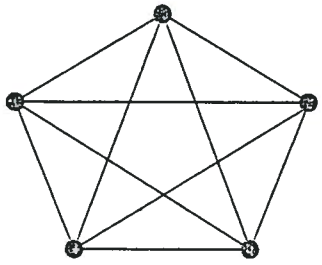
Lemma 9.1.3. *The sum of the degrees of the vertices in a graph equals twice the number of edges.*

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

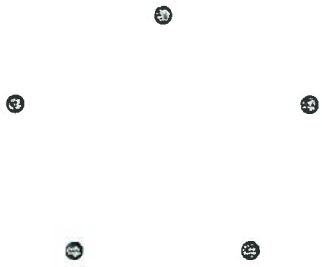
Lemma 9.1.3 is sometimes called the *Handshake Lemma*: if we total up the number of people each person at a party shakes hands with, the total will be twice the number of handshakes that occurred.

9.1.4 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:



The n-node graph containing n-1 edges in sequence is known as the line graph L_n. More formally $L_n = \{V, E\}$ where $V = \{v_1, v_2, \ldots, v_n\}$ and $E = \{\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{n-1}, v_n\}\}$. For example, L_5 is displayed in Figure C.

~~Another 5-vertex graph is L_4, the line graph of length four.~~

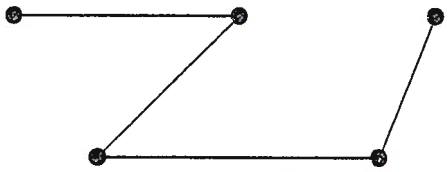


Figure C. The 5-node line graph L_5.

~~The n-node graph~~

~~And here is C_5, a simple cycle with 5 vertices.~~

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

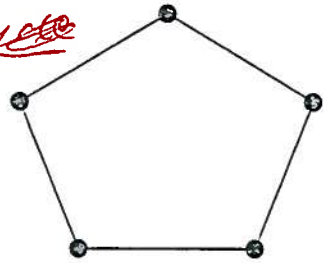


Figure D. The 5-node cyclograph C_5.

9.1.3

~~9.1.5~~ **Isomorphism**

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

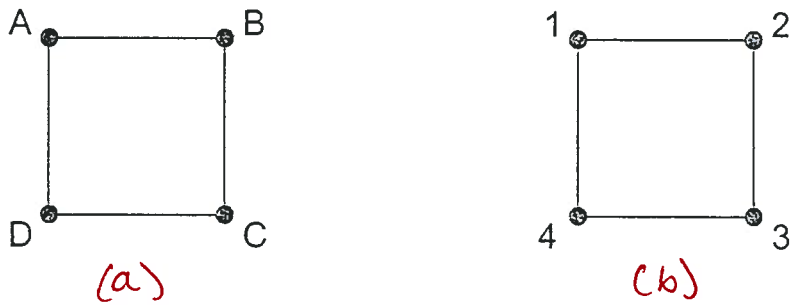


Figure E : Two graphs that are isomorphic to C_4.

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 since the graphs *look the same*!

Fortunately, we can neatly capture the idea of "looks the same" by adapting Definition 6.2.1 of isomorphism of digraphs to handle simple graphs.

Definition 9.1.4. 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 $u, v \in V_1$:

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

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

EDITING NOTE: Graphs G_1 and G_2 are *isomorphic* if there exists a bijection between the vertices in G_1 and the vertices in G_2 such that there is an edge between two vertices in G_1 if and only if there is an edge between the two corresponding vertices in G_2. ■

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

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

For example, here is an isomorphism between vertices in the two graphs shown in Figure 6:

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, we have shown two different ways of drawing C_5 in Figure F.

Figure F: Two ways of drawing C_5.

Isomorphism preserves the connection properties of a graph, abstracting out what the vertices are called, what they are made out of, or where they appear in a drawing of the graph. More precisely, a property of a graph is said to be *preserved under isomorphism* if whenever G has that property, every graph isomorphic to G

also has that property. For example, 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.

9.1.6 Problems

Class Problems

Homework Problems

Exam Problems

9.2 The Stable Marriage Problem

Okay, frequent public reference to derived variables may not help your mating prospects. But they can help with the analysis!

5G.1.4 Subgraphs

~~A graph $G_1 = (V_1$~~

Definition ~~ABC~~ 5D1: A graph $G_1 = (V_1, E_1)$ is said to be a subgraph of a graph $G_2 = (V_2, E_2)$ if $V_1 \subseteq V_2$ and $E_1 \subseteq E_2$.

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

$V = \{H, I\}$ and

$E = \{\{H, I\}\}$

is a subgraph of the graph in Figure AA. On the other hand, ~~the graph~~ any graph containing an edge $\{G, H\}$ would not be a subgraph of the graph in Figure AA because the graph in Figure AA does not contain ~~the~~ this edge.

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

5.1.5 Weighted Graphs

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

In such cases, it is useful to represent the system with an edge-weighted graph, aka, a weighted graph). A weighted graph is the same as a simple graph except that we associate a real number (i.e., the weight) with each edge in the graph. Mathematically speaking, a weighted graph consists of a graph $G = (V, E)$ and a weight function $w: E \to \mathbb{R}$. For example, Figure 5.4 shows a weighted graph where the weight of edge A-B is 5.

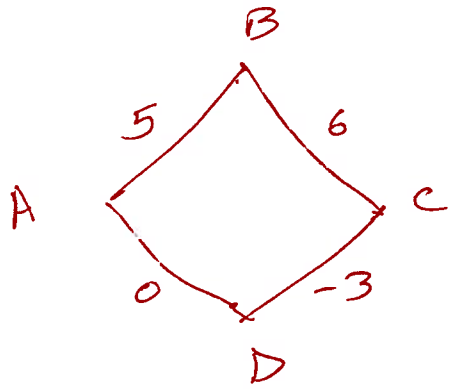


Figure 54: A 4-node weighted graph where edge A-B has weight 5.

5.1.6 Adjacency Matrices

We have already seen two ways: you

There are many ways to represent a graph. A ~~For example, you~~ ~~You~~ can draw it (e.g., as ~~as shown~~ in Figure 54), ~~And~~ and you can represent it ~~as~~ with sets ~~(as in~~ (as in $G = (V, E)$). Another common representation is with an adjacency matrix.

Definition 5AB: Given an n-node ~~matrix~~ graph $G = (V, E)$, the **adjacency matrix** ~~A = [a_ij]~~ where $V = \{v_1, v_2, \ldots, v_n\}$,

for G is the nxn matrix $A_G = \{a_{ij}\}$ where

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

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

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

For example, Figure 5Z displays the adjacency matrices for the graphs shown in Figures E(a) and 5Y where $v_1 = A$, $v_2 = B$, $v_3 = C$ and $v_4 = D$.

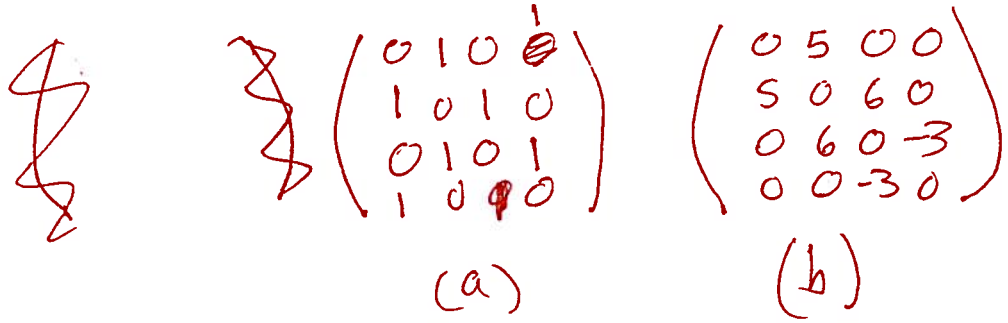


Figure 5Z: Examples of adjacency matrices. (a) shows the adjacency matrix for the graph in Figure E(a) and (b) shows the adjacency matrix for the weighted graph in Figure 5Y. In each case, we set $v_1 = A$, $v_2 = B$, $v_3 = C$ and $v_4 = D$ to construct the matrix.

INSERT 5E

We begin our study of graph theory by considering the scenario where the nodes in a graph represent people and edges represent a relationship between pairs of people such as "likes", "marries", and so on. ~~Although~~ Now, ~~we will spend~~ ~~talk about the~~ ~~the examples will generally be in terms of~~ you may be wondering what marriage has to do with computer science, and with good reason. It turns out that the techniques we will develop ~~can be~~ apply to much more general scenarios ~~that~~ where instead of matching men to women, we need to match packets to paths in a network, ~~or~~ applicants to jobs, or internet traffic to webservers.

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

~~INSERT 5E~~

~~Our Root~~

9.2.1 Sex in America

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

9.2.1 The Problem

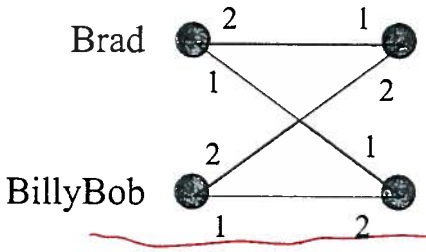
<subsubsection (no #)

We next consider a version of the bipartite matching problem, where there are an equal number of men and women, and where each person has preferences about who they would like to marry. In fact, we ~~Suppose there are a bunch of boys and an equal number of girls that we want to marry off. Each boy has his personal preferences about the girls — in fact, we~~ assume that each man has a complete list of all the women ranked according to his preferences, with no ties. Likewise, each woman has a ranked list of all of the men.

The preferences don't have to be symmetric. That is, Jennifer might like Brad best, but Brad doesn't necessarily like Jennifer best. The goal is to marry everyone ~~off boys and girls~~: every man must marry exactly one woman and vice-versa—no polygamy. ~~In other words, we are looking for a perfect matching.~~ ~~In mathematical terms, we want the mapping from boys to their wives to be a bijection or *perfect matching*. We'll just call this a "matching," for short.~~ Moreover, we would like to find a matching between men and women that is stable in the sense that there is no pair of people that prefer each other to their spouses.

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

This situation is illustrated in the following diagram where the digits "1" and "2" near a boy shows which of the two girls he ranks first and which second, and similarly for the girls.



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

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

Definition 9.2.1. A *stable matching* is a matching with no rogue couples.

The question is, given everybody's preferences, how do you find a stable set

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

It is something of a surprise that there always is a stable matching among a group of men and women, and we'll shortly explain why. The surprise springs in part from considering the apparently similar "buddy" matching problem. That is, if people can be paired off as buddies, regardless of gender, then a stable matching *may not* be possible. For example, Figure 9.2 shows a situation with a love triangle and a fourth person who is everyone's last choice. In this figure Mergatoid's preferences aren't shown because they don't even matter.

Let's see why there is no stable matching:

Lemma. *There is no stable buddy matching among the four people in Figure 9.2.*

Proof. We'll prove this by contradiction.

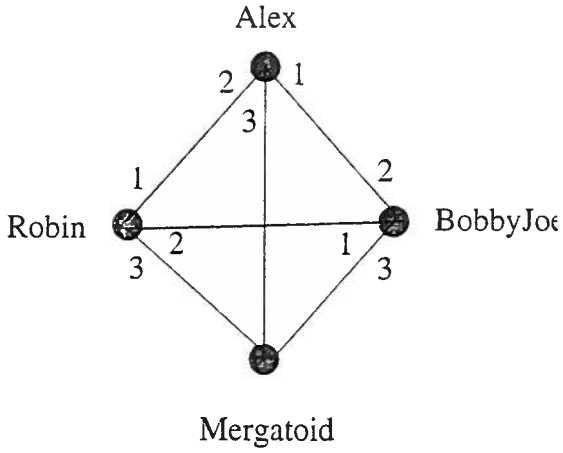


Figure 9.2: Some preferences with no stable buddy matching.

Assume, for the purposes of contradiction, that there is a stable matching. Then there are two members of the love triangle that are matched. Since preferences in the triangle are symmetric, we may assume in particular, that Robin and Alex are matched. Then the other pair must be Bobby-Joe matched with Mergatoid.

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

So getting a stable *buddy* matching may not only be hard, it may be impossible. But when boys are only allowed to marry girls and vice versa, then it turns out

can always be found.[1]

that a stable matching ~~is not hard to find.~~

9.2.2 The Mating Ritual

<subsubsection (no #)

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

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

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

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

Termination condition: ~~When every girl~~ when a day arises in which every woman has at most one suitor, the ritual ends with each woman marrying her suitor, if she has one.

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

[1] once again, we disclaim any political statement here — it's just the way that the math works out.

prove:

- The Ritual has a last day.
- Everybody ends up married.
- The resulting marriages are stable.

9.2.3 A State Machine Model

Before we can prove anything, we should have clear mathematical definitions of what we're talking about. In this section we sketch how to define a rigorous state machine model of the Marriage Problem.

So let's begin by formally defining the problem.

Definition 9.2.2. A *Marriage Problem* consists of two disjoint sets of the same finite size, called the-Boys and the-Girls. The members of the-Boys are called *boys*, and members of the-Girls are called *girls*. For each boy, B, there is a strict total order, $<_B$, on the-Girls, and for each girl, G, there is a strict total order, $<_G$, on the-Boys. If $G_1 <_B G_2$ we say B *prefers* girl G_2 to girl G_1. Similarly, if $B_1 <_G B_2$ we say G

prefers boy B_2 to boy B_1.

A *marriage assignment* or *perfect matching* is a bijection, w : the-Boys $\to$ the-Girls. If $B \in$ the-Boys, then $w(B)$ is called B's *wife* in the assignment, and if $G \in$ the-Girls, then $w^{-1}(G)$ is called G's *husband*. A *rogue couple* is a boy, B, and a girl, G, such that B prefers G to his wife, and G prefers B to her husband. An assignment is *stable* if it has no rogue couples. A *solution* to a marriage problem is a stable perfect matching.

To model the Mating Ritual with a state machine, we make a key observation: to determine what happens on any day of the Ritual, all we need to know is which girls are still on which boys' lists on the morning of that day. So we define a state to be some mathematical data structure providing this information. For example, we could define a state to be the "still-has-on-his-list" relation, R, between boys and girls, where $B\ R\ G$ means girl G is still on boy B's list.

We start the Mating Ritual with no girls crossed off. That is, the start state is the *complete bipartite* relation in which every boy is related to every girl.

According to the Mating Ritual, on any given morning, a boy will *serenade* the girl he most prefers among those he has not as yet crossed out. Mathematically, the girl he is serenading is just the maximum among the girls on B's list, ordered by $<_B$. (If the list is empty, he's not serenading anybody.) A girl's *favorite* is just the maximum, under her preference ordering, among the boys serenading her.

Continuing in this way, we could mathematically specify a precise Mating Ritual state machine, but we won't bother. The intended behavior of the Mating Ritual is clear enough that we don't gain much by giving a formal state machine, so we stick to a more memorable description in terms of boys, girls, and their preferences. The point is, though, that it's not hard to define everything using basic mathematical data structures like sets, functions, and relations, if need be.

9.2.4 There is a Marriage Day

It's easy to see why the Mating Ritual has a terminal day when people finally get married. Every day on which the ritual hasn't terminated, at least one boy crosses a girl off his list. (If the ritual hasn't terminated, there must be some girl serenaded

by at least two boys, and at least one of them will have to cross her off his list).

So starting with n boys and n girls, each of the n boys' lists initially has n girls on it, for a total of n^2 list entries. Since no girl ever gets added to a list, the total number of entries on the lists decreases every day that the Ritual continues, and so the Ritual can continue for at most n^2 days.

9.2.3 They All Live Happily Every After...

We still have to prove that the Mating Ritual leaves everyone in a stable marriage. To do this, we note one very useful fact about the Ritual: if a girl has a favorite boy suitor on some morning of the Ritual, then that favorite suitor will still be serenading her the next morning —because his list won't have changed. So she is sure to have today's favorite boy among her suitors tomorrow. That means she will be able to choose a favorite suitor tomorrow who is at least as desirable to her as today's favorite. So day by day, her favorite suitor can stay the same or get better, never worse. In others words, a girl's favorite is a weakly increasing variable with respect to her preference order on the boys.

Now we can verify the Mating Ritual using a simple invariant predicate, P, that captures what's going on:

Let P be the predicate:

For every woman, w, and every man, m, if w is crossed off m's list, then w has a suitor whom she prefers over m.

INSERT P goes here

Why is P invariant? Well, we know that w's favorite tomorrow will be at least as desirable to her as her favorite today, and since her favorite today is more desirable than m, tomorrow's favorite will be too.

Notice that P also holds on the first day, since every woman is on every list. So by the Invariant Theorem, we know that P holds on every day that the Mating Ritual runs. Knowing the invariant holds when the Mating Ritual terminates will let us complete the proofs.

In other words, no one has been crossed off any list and P is vacuously true. Hence P is means that P is an invariant and thus it must hold on the day when the Ritual terminates.

Theorem 9.2.3. *Everyone is married by the Mating Ritual.*

By Contradiction. Assume

Proof. Suppose, for the sake of contradiction, that it is the last day of the Mating Ritual and some boy does not get married. Then he can't be serenading anybody, and so his list must be empty. So by invariant P, every woman has a suitor who is a favorite boy.

since there are an equal someone

since there

INSERT Q goes her

INSERT P

Lemma 5P: P is an invariant for the Mating Ritual.

Proof: By induction on the number of days.

Base case: ~~On the first day~~ ~~At the~~ at the end of day 0), in the beginning (i.e., ~~day 0~~) every woman is on every list — no one has been crossed off and so P is vacuously true.

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

Case 1: w was crossed off m's list on day d+1. Then w must have a suitor she ~~was only because~~ ~~w has a suitor~~ prefers on day d+1.

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

she prefers better at the end of day d+1.

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

~~Using Lemma~~

With Lemma 5 P in hand, we can now prove:

~~Thm~~

INSERT Q

number of ~~BOBS~~ men and women, and since bigamy is not allowed, this means that at least one man (call him ~~Bob~~ Bob) and at least one woman do not get married.

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

whom she prefers to that boy. In particular, every girl has a favorite boy whom she marries on the last day. So all the girls are married. What's more there is no bigamy: a boy only serenades one girl, so no two girls have the same favorite. But there are the same number of girls as boys, so all the boys must be married too. ■

Theorem 9.2.4. *The Mating Ritual produces a stable matching.*

Proof. Let Brad be some boy and Jen be any girl that he is *not* married to on the last day of the Mating Ritual. We claim that Brad and Jen are not a rogue couple. Since Brad is an arbitrary boy, it follows that no boy is part of a rogue couple. Hence the marriages on the last day are stable.

To prove the claim, we consider two cases.

Case 1. Jen is not on Brad's list. Then by invariant P, we know that Jen prefers her husband to Brad. So she's not going to run off with Brad, the claim holds in this case.

Case 2. Otherwise, Jen is on Brad's list. But since Brad is not married to Jen, he

must be choosing to serenade his wife instead of Jen, so he must prefer his wife.

So he's not going to run off with Jen. ■

9.2.6 ...Especially the Boys

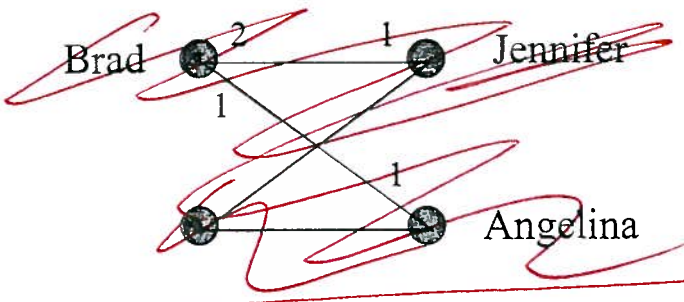
Who is favored by the Mating Ritual, the boys or the girls? The girls seem to have all the power: they stand on their balconies choosing the finest among their suitors and spurning the rest. What's more, we know their suitors can only change for the better as the Ritual progresses. Similarly, a boy keeps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

But it's not! The fact is that from the beginning, the boys are serenading their first choice girl, and the desirability of the girl being serenaded decreases only enough to give the boy his most desirable possible spouse. The mating algorithm actually does as well as possible for all the boys and does the worst possible job

for the girls

To explain all this we need some definitions. Let's begin by observing that while the mating algorithm produces one stable matching, there may be other stable matchings among the same set of boys and girls. For example, reversing the roles of boys and girls will often yield a different stable matching among them.

But some spouses might be out of the question in all possible stable matchings. For example, Brad is just not in the realm of possibility for Jennifer, since if you ever pair them, Brad and Angelina will form a rogue couple; here's a picture:



Definition 9.2.5. Given any marriage problem, one person is in another person's *realm of possible spouses* if there is a stable matching in which the two people are married. A person's *optimal spouse* is their most preferred person within their realm of possibility. A person's *pessimal spouse* is their least preferred person in their

realm of possibility.

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

Theorem 9.2.6. *The Mating Ritual marries every boy to his optimal spouse.*

Proof. Assume for the purpose of contradiction that some boy does not get his optimal girl. There must have been a day when he crossed off his optimal girl —otherwise he would still be serenading her or some even more desirable girl.

By the Well Ordering Principle, there must be a *first* day when a boy is called him "Keith") crosses off his optimal girl, Nicole.

According to the rules of the Ritual, Keith crosses off Nicole because Nicole has a favorite suitor, Tom, and

Nicole prefers Tom to Keith (*)

(Remember, this is a proof by contradiction).

Now since this is the first day an optimal girl gets crossed off, we know Tom

had not previously crossed off his optimal spouse, and so ~~hasn't crossed off his optimal girl. So~~

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

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

Theorem 9.2.7. *The Mating Ritual marries every ~~girl~~ woman to her pessimal spouse.*

By contradiction. Assume the theorem is not true.

Proof. ~~Say Nicole and Keith marry each other as a result of the Mating Ritual. By the previous Theorem 9.2.6, Nicole is Keith's optimal spouse, and so in any stable set of marriages,~~ he only proposed to her

Keith rates Nicole at least as high as his spouse. (+)

Now suppose for the purpose of contradiction that there is another stable set of marriages where Nicole does worse than Keith. That is, Nicole is married to Tom, and

INSERT R goes here

INSERT R

~~and let Nicole~~

Hence there must be a stable set of marriages M where some woman (call her Nicole) is married to ~~some~~ a man (call him Tom) that she likes less than her spouse in the Mating Ritual (call him Keith). This means that

Nicole prefers Keith to Tom (+).

By Theorem 9.2.6 and the fact that Nicole and Keith are ~~matched~~ married in the ~~Mating~~ Ritual, we know that

Keith prefers Nicole to ~~another~~ his spouse, in M (++).

This means that Keith and Nicole form a rogue couple in M, which contradicts the stability of M. ∎

Nicole prefers Keith to Tom (++).

Then in this stable set of marriages where Nicole is married to Tom, (+) and (++) imply that Nicole and Keith are a rogue couple, contradicting stability. We conclude that Nicole cannot do worse than Keith. ■

9.2.1 Applications

Not surprisingly, a stable matching procedure is used by at least one large dating agency. But although "boy-girl marriage" terminology is traditional and makes some of the definitions easier to remember (we hope without offending anyone), solutions to the Stable Marriage Problem are widely useful.

The Mating Ritual was first announced in a paper by D. Gale and L.S. Shapley in 1962, but ten years before the Gale-Shapley paper was appeared, and unknown by them, the Ritual was being used to assign residents to hospitals by the National Resident Matching Program (NRMP). The NRMP has, since the turn of the twentieth century, assigned each year's pool of medical school graduates to hospital

residencies (formerly called "internships") with hospitals and graduates playing the roles of boys and girls. (In this case, there may be multiple boys married to one girl, but there's an easy way to use the Ritual in this situation (see Problem ??).

Before the Ritual was adopted, there were chronic disruptions and awkward countermeasures taken to preserve assignments of graduates to residencies. The Ritual resolved these problems so successfully, that it was used essentially without change at least through 1989.[2]

The internet infrastructure company, Akamai, also uses a variation of the Gale-Shapley procedure to assign web traffic to servers. In the early days, Akamai used other combinatorial optimization algorithms that got to be too slow as the number of servers (over 20,000 in 2010) and traffic increased. Akamai switched to Gale-Shapley since it is fast and can be run in a distributed manner. In this case, the web traffic corresponds to the boys and the web servers to the girls.

[2] Much more about the Stable Marriage Problem can be found in the very readable mathematical monograph by Dan Gusfield and Robert W. Irving, **The Stable Marriage Problem: Structure and Algorithms**, MIT Press, Cambridge, Massachusetts, 1989, 240 pp.

~~girls~~ The web requests ~~The servers~~ have preferences based on latency and packet loss; ~~the traffic has~~ and the web servers have preferences based on the cost of bandwidth and colocation.

9.2.8 Problems

Practice Problems

Class Problems

Homework Problems

EDITING NOTE: Add problem proving that the Mating Ritual need not proceed in morning/afternoon/evening lock step: a girl can reject nonfavorite suitors one at a time and at any time, and a rejected boy can change the girl he serenades without waiting for the other boys to change. The proof uses the fact that single actions commute, so induction proves that all executions are confluent —which implies all executions end with the same boy-optimal matching. This lemma can be cited in the planar graphs section to prove that the edges in an embedding can be added in any order. ∎

—INSERT 5 (old section 9.5) goes here on pp414–422—

9.3 Connectedness

5.4 Getting From A to B in a Graph

9.3.1 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 7.1.1 of paths in digraphs:

Definition 9.3.1. A *path* in a graph, G, is a sequence of $k+1$ 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*[3] iff all the v_i's are different, that is,

[3] 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".

$i \neq j$ implies $v_i \neq v_j$.

For example, the graph in Figure 9.3 has a length 6 simple path A,B,C,D,E,F,G. This is the longest simple path in the graph.

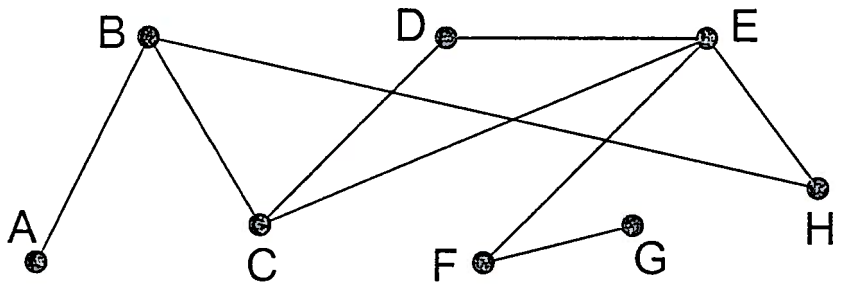


Figure 9.3: *A graph* with 5 single cycles.

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.

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 9.3. 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 9.3 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. [4]

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 9.3.2. 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' must be vertices in V'.

Definition 9.3.3. For $n \geq 3$, let C_n be the graph with vertices $1, \ldots, n$ and edges

$$1\text{—}2, \quad 2\text{—}3, \quad \ldots, \quad (n-1)\text{—}n, \quad n\text{—}1.$$

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.

[4] 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.

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

INSERT 5AC goes here (it is old sec 9.3.4)

~~9.3.2 Connected Components~~

INSERT 5AD goes here (new text)

5.5 Connectivity

Definition 9.3.4. 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.

INSERT 5AE goes here

EDITING NOTE:

Now if there is a path from vertex u to vertex v, then v is connected to u by the reverse path, so connectedness is a symmetric relation. Also, if there is a path from u to v, and also a path from v to w, then these two paths can be combined to form a path from u to w. So the connectedness relation is transitive. It is also reflexive, since every vertex is by definition connected to itself by a path of length zero.

INSERT 5AE moves up to here

INSERT 5AF goes here

The diagram in Figure 9.5 looks like a picture of three graphs, but is intended

5.4.3 Numbers of Paths

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

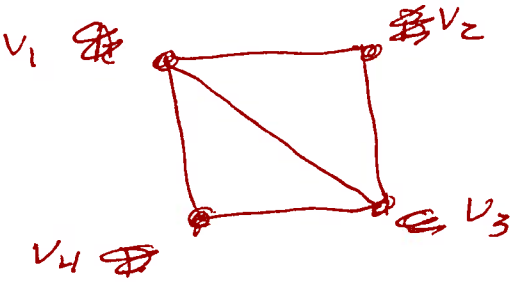


Figure AD. A graph in which there are 5 paths of length 3 from v_1 to v_4. The paths are (v_1, v_2, v_1, v_4), (v_1, v_3, v_1, v_4), (v_1, v_4, v_1, v_4), (v_1, v_2, v_3, v_4), and (v_1, v_4, v_3, v_4).

There is a surprising relationship between the number of paths of length k between a pair of nodes in a graph G and the kth power of the adjacency matrix A_G for G. The relationship is captured in the following

Theorem.

Theorem 5AE: Let $G = (V, E)$ be an n-node graph with $V = \{v_1, v_2, \ldots, v_n\}$, and let $A_G = \{a_{ij}\}$ denote the adjacency matrix for G. Let $a_{ij}^{(k)}$ denote the (i,j)-entry of the kth power of A_G. Then the number of paths of length k between v_i and v_j is $a_{ij}^{(k)}$.

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

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

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

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

Sure enough, ~~$a_{14}^{(3)}$~~ the $(1,4)$ coordinate of A^3 is $a_{14}^{(3)} = 5$, which ~~is~~ is the ~~number~~ number of length 3 paths from v_1 to v_4. And, $a_{24}^{(3)} = 2$, which is the ~~number~~ number of length 3 paths from v_2 to v_4. ~~Let's~~ By proving the theorem, we'll discover why it is true and thereby uncover the relationship between matrix multiplication and numbers of paths.

Proof of Theorem 5AE: The proof is by induction on k. We will let $P(k)$ be ~~let $P(k)$~~ the predicate that the theorem is true for k.

~~For paths of length k. For k, we will denote the number of paths between v_i and v_j of length k between v_i and v_j.~~

~~Base case: $(k = 1)$.~~ ~~There is a path of length $k = 1$ between v_i and v_j iff there is an edge connecting v_i and v_j.~~ This is true iff $a_{ij} = a_{ij}^{(1)} = 1$. There can be at most one path of length 1 between v_i and v_j. There are 0 paths between v_i and v_j iff ~~the~~ $\{v_i, v_j\} \notin E$, which is ~~true iff~~

↳ The goal is to show

Let $P_{i,j}^{(k)}$ denote the number of paths of length k between v_i and v_j. The P(k) is the predicate:

$$\forall i, j \in [1, n]. \quad P_{ij}^{(k)} = a_{ij}^{(k)}. \quad \text{(eqn 5AE)}$$

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

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

Case 2: $\{v_i, v_j\} \notin E$. Then $P_{ij}^{(1)} = 0$ since there cannot be any paths of length 1 between v_i and v_j. Moreover, $\{v_i, v_j\} \notin E$ means that $a_{ij} = 0$. So $P_{ij}^{(1)} = a_{ij}^{(1)}$ in this case as well. ☑

Hence, P(1) must be true.

Inductive Step: Assume P(k) is true. In other words, assume that Equation 5AE holds.

We can count the number of paths of length $k+1$ from v_i to v_j according to the ~~next~~ first ~~node visited after~~ ~~(call it v_t)~~ ~~after~~ ~~edge~~ edge in the path (call it $\{v_i, v_t\}$). This means that

$$P_{i,j}^{(k+1)} = \sum_{t:\{v_i,v_t\}\in E} P_{t,j}^{(k)} \qquad \text{Eqn AF}$$

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

$$P_{i,j}^{(k+1)} = \sum_{t=1}^{n} a_{it} P_{t,j}^{(k)} .$$

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

$$P_{i,j}^{(k+1)} = \sum_{t=1}^{n} a_{i,t} a_{t,j}^{(k)} .$$

But the formula for matrix multiplication gives that

$$a_{i,j}^{(k+1)} = \sum_{t=1}^{n} a_{i,t}\, a_{t,j}^{(k)}$$

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

5.4.4 Shortest Paths

Although the connection between the power of the adjacency matrix and the number of paths is cool (at least ~~if you~~ ~~are cool to~~ if you are a mathematician), ~~counting~~ the problem of counting paths does not come up very often in practice. Much more important is the problem of finding the shortest path ~~in a graph~~ between a ~~pair~~ pair of nodes in a graph. There is good news and bad news to report on this front.

The good news is that it is not very hard to find a shortest path. ~~There are several ways to do it~~ The bad news is that you can't win one of those million Dollar prizes for doing it.

In fact, there are several good algorithms known for finding ~~the~~ a shortest path between a pair of nodes, ~~The simplest to explain~~ ~~In fact, one way~~ (but not ~~the best~~ the best) is to compute the powers of the adjacency matrix one by one until the value of $a_{ij}^{(k)}$ exceeds 0. ~~That's because The smallest value by~~ Theorem 5AE2 implies that the length of the shortest path between v_i and v_j will be the smallest value of k for which $a_{ij}^{(k)} > 0$.

~~Weighted~~ Paths in Weighted Graphs ← subsubsection (no #)

~~The Scent~~

The problem of computing shortest paths in a weighted graph frequently arises in practice. For example, when you drive home

for vacation, you usually would like to take the shortest route. ~~In the case of a weighted graph, the length of a path is simply the sum of the weights~~

Definition 5H: Given a weighted graph, the length of a path in the graph is the sum of the weights of the edges in the path.

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

~~DATES~~

INSERT 5AF

5.5.1 Connected components

~~Connectivity~~

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~~ where nodes represent cities and ~~edges~~ edges represent highways might be connected for North American cities, but would ~~not~~ 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 ~~immune~~ protected from viruses that spread on the Internet, some government networks are completely isolated from the Internet.

For example, the

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.

EDITING NOTE:

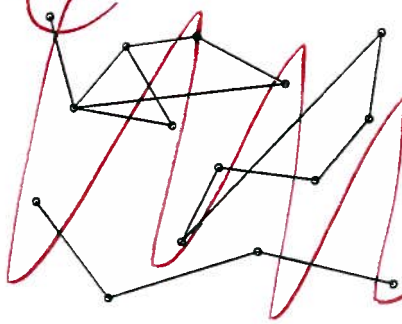


Figure 9.4: *One graph with 3 connected components.*

Definition 9.3.5. 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 rigor-~~ ~~ous definition is easy:~~ A connected component ~~is the set of all the vertices connected~~ is a subgraph of a graph consisting of some vertex and every node and egde that is connected to that vertex. ~~by a path.~~

Figure 9.5: *One graph with 3 connected components.*

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.

9.3.2 k-connected graphs

9.3.3 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 9.3.6. 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[5] 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 9.3 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

[5]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."

case $k = 2$.

This is insert SAC and moves after sections 4 & 5

5.4.2 Finding the

~~9.3.4 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 9.3.7. *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

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

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

5.5.3

~~9.3.5~~ The Minimum Number of Edges in a Connected Graph

~~In order to be~~

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

Theorem 9.3.9. *Every graph with v vertices and e edges has at least $v - e$ connected components.*

Of course for Theorem 9.3.9 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 v vertices. We want to prove that G has at least $v - (e + 1)$ connected components. To do this, remove an arbitrary edge a—b and call the resulting graph G'. By the induction assumption, G' has at least $v - e$ connected components. Now add back the edge a—b to obtain the

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

Corollary 9.3.10. *Every connected graph with v vertices has at least $v - 1$ edges.*

A couple of points about the proof of Theorem 9.3.9 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 ?? and ??. Always use shrink-down, grow-back arguments, and you'll never fall into this trap.

INSERT 5AG goes here

9.3.6 Problems

INSERT 5AH goes here

Class Problems

INSERT 5AI goes here
(text from end of old Section 9.4 on pp 411-)

Homework Problems

INSERT 5AJ goes here

Homework Problems

INSERT 5AK goes here
(text from p 413)

INSERT 5AL goes here

9.4 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

INSERT 5A6

5A6-1

6.042/18.062J Mathematics for Computer Science
Tom Leighton, Marten van Dijk, and Brooke Cowan

September 30, 2008
Lecture Notes

Graph Theory III

5.5.4

1 Build-Up Error

Here is a false proof about connectivity. It exposes a very common flaw made on proofs by induction on graphs – it even has a name – it is known as "build-up error".

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

There are many counterexamples; ~~here is one:~~ for example, see Figure 5Z.

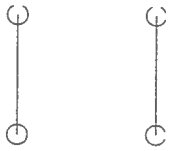


Figure 5Z: A counterexample graph to the False claim.

Since the claim is false, there must be at least one error in the following "proof".

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

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

Inductive step: We must show that $P(n)$ implies $P(n+1)$ for all $n \geq 1$. Consider an n-vertex graph in which every vertex has degree at least 1. By the assumption $P(n)$, this graph is connected; that is, there is a path between every pair of vertices. Now we add one more vertex x to obtain an $(n+1)$-vertex graph as shown in Figure 5Y.

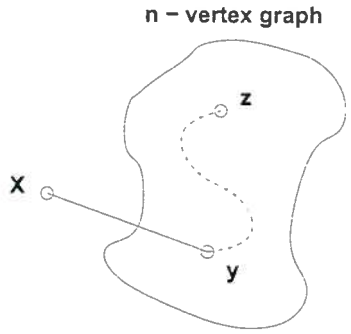


Figure 5Y: Adding a vertex x with degree at least 1 to a connected n-node graph.

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

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

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

The counterexample above shows that this assumption is false; there is no way to build the 4-vertex graph from a 3-vertex graph with minimum degree 1. Thus the first error in the proof is the statement "This proves $P(n+1)$".

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

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

Inductive step: We must show that $P(n)$ implies $P(n+1)$ for all $n \geq 1$. Consider an $(n+1)$-vertex graph G in which every vertex has degree at least 1. Remove an arbitrary vertex v, leaving an n-vertex graph G' in which every vertex has degree... uh-oh!

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

2 Trees

Today we'll talk about a very special class of graphs called *trees*. Trees arise in all sorts of applications and you'll see them in just about every computer science class that you'll take at MIT. There are at least half a dozen ways to define a tree, but the simplest is the following.

Definition. *A* tree *is a simple*[1]*, connected, acyclic graph.*

[1] Recall that we only consider simple graphs in this class, that is, graphs without loops or multiple edges.

5.6 Around and Around We Go!

5.6.1 Cycles

Definition[1]: Given a graph $G=(V,E)$, a *cycle* in G is a sequence of nodes $v_0, v_1, \ldots, v_k$ where $v_i - v_{i+1} \in E$ for $0 \le i < k$ and $v_0 = v_k$. The *length* of the cycle is k. A cycle is a *simple cycle* if $k \ge 3$ and $v_0, v_1, \ldots, v_{k-1}$ are all different.

For example, B, C, D, E, C, B is a cycle in the graph shown in Figure 3, (of length 5) it is not simple since it contains node C twice. There are many ways to represent a cycle. For

---

1 Some texts use the term *closed walk* or *circuit* instead of cycle and *cycle* instead of simple cycle.

example, B, C, D, E, C, B is the same as C, D, E, C, B, C (just starting at node C instead of node B) and the same as B, C, E, D, C, B (just reversing direction).

Cycles are very similar to paths, except that the last node is the first node and the notion of first and last does not matter.

5.6.2 Euler Tours

INSERT 5AJ

It is not difficult to extend Theorem 9.4.3 to prove that a connected graph, G, has an Euler path if and only if precisely 0 or 2 nodes in G have odd degree. Hence, we can conclude that the graph shown in Figure 5BC has an Euler path but not an Euler tour since the graph has precisely two nodes with odd degree.

Although the proof of Theorem 9.4.3 does not explicitly define a method for finding an Euler tour when one exists, it is not hard to modify the proof to produce such a method. The idea is to grow a tour by continually splicing in cycles until all the edges are consumed.

5.6.4 The Traveling Sales[illegible] and The

5.6.3 Hamiltonian Cycles and Other

The Traveling Salesman Problem

Traveling Salesperson Problem

Hamiltonian cycles are the unruly cousins of Euler tours.

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

Although Hamiltonian cycles sound similar to Euler tours — one visits every node once while the other visits every edge once — finding a Hamiltonian cycle can be a lot harder than finding an Euler tour. The same is true for Hamiltonian paths. This is because no one has discovered a

5.6.4 The ~~Travelling~~ ~~Salesman~~ person ~~Problem~~

5.6.4 The Travelling Salesperson Problem

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

Definition:

~~Definition~~: Given a weighted graph G, the weight of a cycle in G is defined to be the sum of the weights of the ~~edges~~ in the cycle. ~~The Travelling Salesman Problem is the problem of finding a Hamiltonian cycle~~

For example, consider the graph shown in Figure AL and suppose that you would like to visit every node once and finish at the node where you started. Can you find a way to do this by traversing a cycle with weight 15 ?

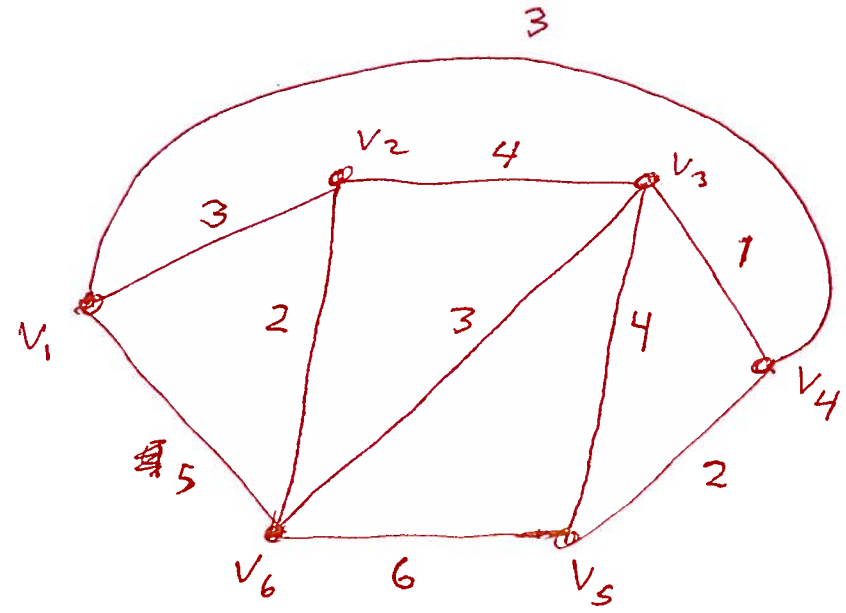


Figure AL : A weighted graph. Can you find a cycle that visits every node exactly once with weight 15?

~~A Perhaps it is~~

A Needless to say, if you can figure out a fast procedure ~~to always~~ that finds ~~the minimum weight~~ the optimal ~~cyc~~ cycle for the ~~the~~ travelling salesman, you can win a million dollars.

~~Let us know~~

5.7 Trees

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

Definition 5.7.1: A connected acyclic graph is called a tree.

Trees are a fundamental data structure in computer science and you will encounter them in every CS course that you take.

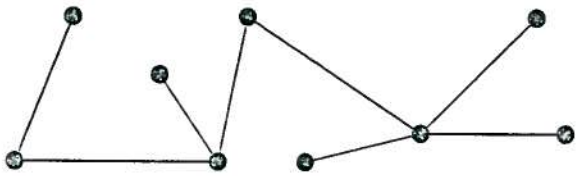
purest kind of tree. Namely, we use the term *tree* to mean a connected graph without simple cycles.

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

5.7.1

9.4.1 Tree Properties (STILL TO BE EDITED)

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 9.4.1. *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.*

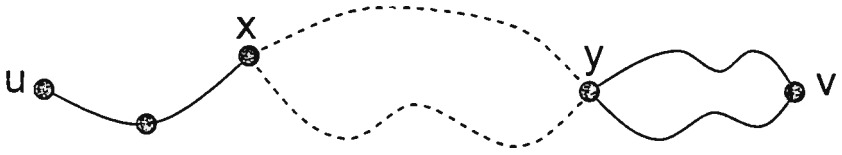
4. *Removing any edge disconnects the graph.*

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

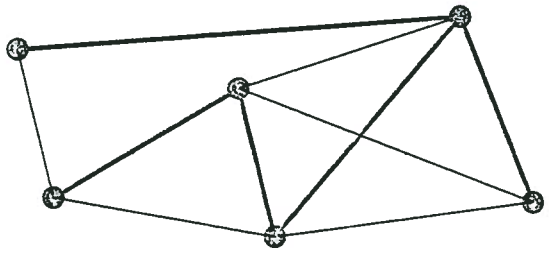
vertices and number of edges both increase by 1. Thus, the claim holds for T and, by induction, for all trees.

■

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.

9.4.2 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 9.4.2. *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:

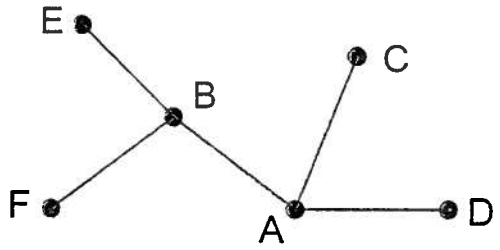
$$v_0—v_1, v_1—v_2, \ldots, v_n—v_0$$

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. ∎

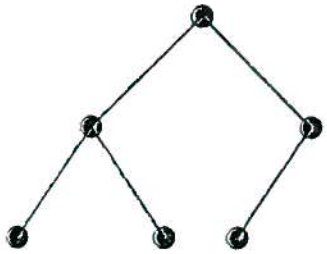
EDITING NOTE:

Tree Variations

Trees come up often in computer science. For example, information is often stored in tree-like data structures and the execution of many recursive programs can be regarded as a traversal of a tree. There are many varieties of trees. For example, a *rooted tree* is a tree with one vertex identified as the *root*. Let u—v be an edge in a rooted tree such that u is closer to the root than v. Then u is the *parent* of v, and v is a *child* of u.

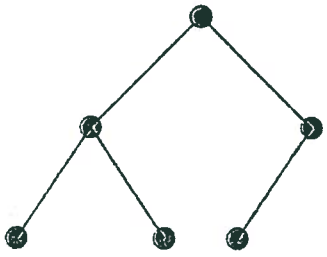


In the tree above, suppose that we regard vertex A as the root. Then E and F are the children of B, and A is the parent of B, C, and D. A *binary* tree is a rooted tree in which every vertex has at most two children. Here is an example, where the top-



most vertex is the root.

In an *ordered, binary* tree, the children of a vertex v are distinguished. One is called the *left child* of v, and the other is called the *right child*. For example, if we regard the two binary trees below as unordered, then they are equivalent. However, if we regard these trees as ordered, then they are different.



9.7.3 Minimum Weight Spanning Trees (TBA)

EDITING NOTE:

Problem ?? presents most of this as a homework problem.

5.8 Planar Graphs

CHAPTER 11 will go here when I edit it

5.9 Problems

SAI - 1

and goes top 402

Dave: take out of staff notes mode

THIS IS INSERT SAI

Traversing a Graph

Can you walk every hallway in the Museum of Fine Arts *exactly once*? If we represent hallways and intersections with edges and vertices, then this reduces to a question about graphs. For example, could you visit every hallway exactly once in a museum with the floorplan shown in Figure 5BC?

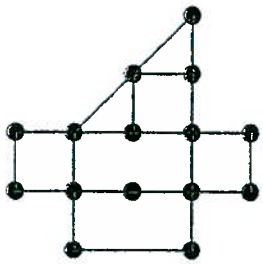


Figure 5BC: A possible floorplan for a museum. Can you find a cycle that traverses every edge exactly once?

Euler Tours and Hamiltonian Cycles

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

traverses every edge exactly once. Graphs with Euler tours and Euler walks both have simple characterizations.

Theorem 9.4.3. *A graph has an Euler tour iff it is connected and every vertex has even degree.*

Proof. Suppose a graph has an Euler tour. Every pair of vertices must appear in the tour, so the graph is connected. Moreover, a vertex that appears k times in the tour must have degree $2k$, so every vertex of the graph has even degree.

Conversely, suppose every vertex in a graph, G, has even degree. Let $W =$ be the longest path in G that traverses every edge at most once. Now W must traverse every edge incident to v_n; otherwise, the path could be extended. In particular, the W traverses two of these edges each time it passes through v_n, and it traverses v_{n-1} — v_n at the end. This accounts for an odd number of edges, but the degree of v_n is even by assumption. Therefore, the W must also begin at v_n; that is, $v_0 = v_n$. Suppose that W is not an Euler tour. Because G is a connected

INSERT 5CC

Proof: We first show that if a graph has an Euler tour, then every vertex has even degree.

Assume that a graph $G = (V, E)$ has an Euler tour $v_0, v_1, \ldots, v_K$ where $v_K = v_0$. Since every edge is traversed once in the tour, $K = |E|$, and the degree of a node u in G is twice the number of times that node appears in the sequence $v_0, v_1, \ldots, v_{K-1}$ times two. We multiply by two since every time the node appears in the sequence, there is an i where $0 < i < K$, then both $v_{i-1} - v_i$ and $v_i - v_{i+1}$ are edges incident to u in G. If $i = 0$, then v_i is replaced by v_{K-1} in the sequence. If $u = v_0 = v_K$, then both $v_{K-1} - v_K$ and $v_0 - v_1$ are edges incident to u in G. Note that no edge is ever counted twice in this argument. Hence, the degree of every node is even.

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

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

graph, we can find an edge not in W but incident to some vertex in W. Call this edge u—v_i. But then we can construct a longer walk

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

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

Corollary 9.4.4. *A connected graph has an Euler walk if and only if either 0 or 2 vertices have odd degree.*

Hamiltonian cycles are the unruly cousins of Euler tours. A *Hamiltonian cycle* is walk that starts and ends at the same vertex and visits every vertex in a graph exactly once. There is no simple characterization of all graphs with a Hamiltonian cycle. In fact, determining whether a given graph has a Hamiltonian cycle is in the same category of problem as the SAT problem of Section 1.4: you get a million dollars for finding an efficient way to determine when a graph has a Hamiltonian cycle —or for proving that no procedure works efficiently on all graphs.

This is INSERT 5AK and goes to page 402

S-1

THIS is Insert S and it becomes Section 5.3

9.4.3 Problems

Class Problems

Homework Problems

5.3 ~~9.3~~ Coloring Graphs, Coloring

In ~~section 9.1~~ 9.2, we used edges to indicate an affinity between ~~two nodes, but have~~ a pair of nodes. We now consider situations where ~~we need~~ it is useful to use ~~ing an edge~~ edges to represent a *conflict* between ~~two nodes~~ a pair of nodes. ~~also turns out to be really useful.~~ For example, consider the following exam scheduling problem.

~~9.6 Modelling Scheduling Conflicts~~

5.3.1 ~~E.S.T~~ 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. 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:

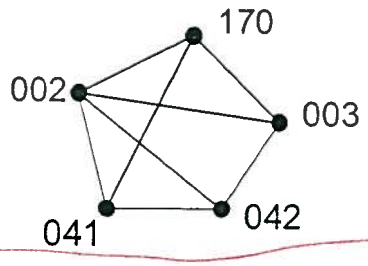


Figure 5R: A scheduling graph for five courses. Exams connected by an edge cannot be given at the same time.

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 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 55, for our example, three colors suffice:

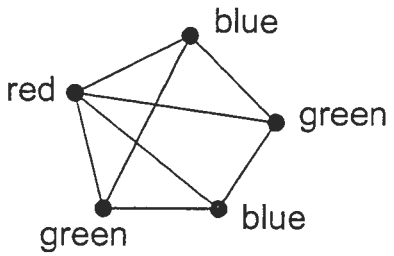


Figure 55: A 3-coloring of the exam graph from Figure 5R.

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 9.6.1. 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!)

9.6.1 Degree-bounded Coloring

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

Theorem 9.6.2. *A graph with maximum degree at most k is $(k+1)$-colorable.*

The natural way to try to prove this theorem is to use induction on k. Unfortunately, this approach leads to disaster. It is not that

it is impossible, just that it is extremely painful and would ruin 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 $\chi(K_n) = n$. In this case, every node has degree $k = n-1$ and so this is an So K_{k+1} is an example where Theorem 9.6.2 gives the best possible bound. This By a example where means that Theorem 9.6.2 gives the best possible bound for *any* graph with Similar argument, we can show degree bounded by k that has K_{k+1} as a subgraph.

EDITING NOTE:

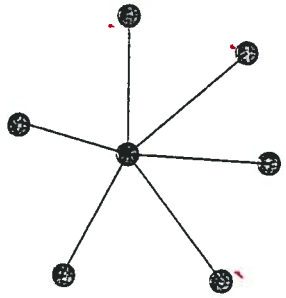
The complete graph, K_n, is also called a size n *clique*, just like a clique of friends, where nodes represent the people and an edge represents the friendship relationship.[6]

For example,

But sometimes $k+1$ colors is far from the best that you can do. Here's an exam-

[6] When speaking of friends, clique is usually pronounced similar to click. However, for some reason, graph theorists think that the word clique rhymes with geek.

The n-node star graph shown in Figure 5T has maximum degree n-1 but can be colored using just 2 colors.



~~ple of an n-node star graph for $n = 7$:~~

Figure 5T: A 7-node star graph.

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

5,3,3

~~9.6.2~~ 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~~ 65,000+ servers every few days. The updates cannot be done at the same time since the servers need to be taken down in order to deploy the software. Also, the servers cannot be handled one at a time, since it would take forever to update them all (each one takes about an hour). Moreover, certain pairs of servers cannot be taken down at the same time since they have common critical functions.

This problem was eventually solved by making a 23,000 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 1.5.4. The question is how many colors are needed to color a map so that adjacent ter-

ritories 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). Surprisingly, ~~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. (It's ~~But it's~~ always easy to tell if an *arbitrary* graph is 2-colorable.) ~~as we show in Section 9.7~~. Later in this chapter, ~~Chapter 11~~, we'll develop enough planar graph theory to present an easy proof ~~at least~~ that all planar graphs are 5-colorable.

H-1

9.6.3 Problems

Class Problems

Homework Problems

Exam Problems

DAVID:
The material in the following section is INSERT H and goes to page 371.

~~9.7~~ 5.2.2 Bipartite Matchings ← now a subsection

~~9.7.1~~ Bipartite Graphs ← now a subsubsection (no #)

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

Definition 9.7.1. 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:~~ the graph in Figure 9.1.

— INSERT J goes here ——

INSERT J

J-1
H-2

The Problem ⊂ subsubsection (no #)

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

But what about getting a mate for every men so that every woman is paired with at most one man? Is it possible to do this so that each man is paired with a woman that he likes? The answer, of course, depends on the bipartite graph that represents who likes who, but the good news is that it is possible to ~~characterize the answer to this question~~ in terms of natural

~~the who-likes-who graph~~

find natural properties of the who-likes-who graph that completely determine the answer to this question.

In general, suppose that we have a set of men and an equal-sized or larger set of women, and there is a graph with an edge between a man and a woman if the man likes the woman. (Note that in this scenario, the "likes" relationship need not be symmetric, since for the time being, we will only worry about finding a mate for a man that he likes.[1] Later, we will consider the "likes" relationship from the female perspective as well.) For example, we might obtain the graph in Figure 5J.

read too much into this example, ~~all this too take offense seriously.~~

[1] By the way, we do not mean to imply that ~~position on~~ mates or pairs should or should not be of a heterosexual nature. ~~Sh~~ Nor do we mean to imply that men should get their choice instead of women. It's just that with bipartite graphs, the edges only connect male nodes to female nodes and there are fewer men in America. ~~with that out of~~ So please don't ~~sue us.~~

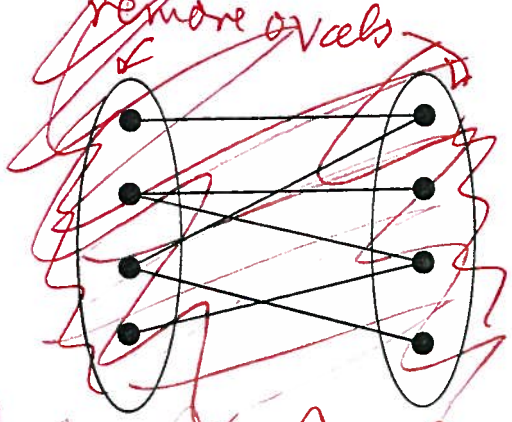


Figure 5a: An example of a bipartite graph

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

The proof of Theorem 9.7.2 is left to Problem ??.

9.7.2 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:

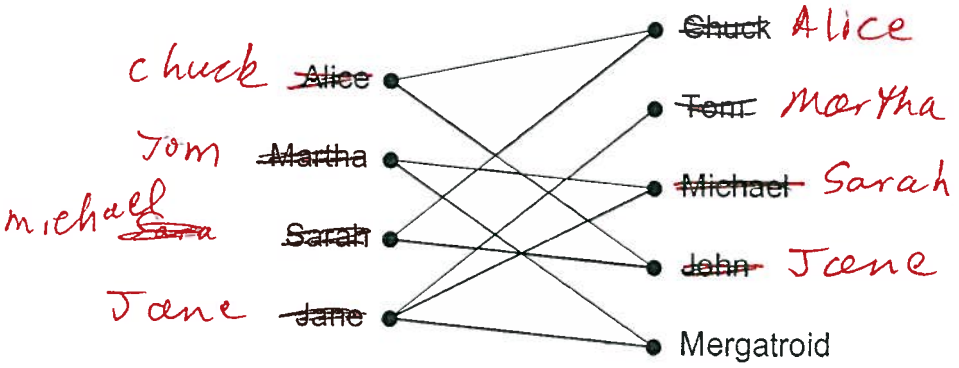


Figure 5.J: A graph where an edge between a man and a woman denotes that the man likes the woman.

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

H-6

likes. For example, here is one possible matching for the girls. men is shown in Figure 5K.

David: we'll need to illustrate the matching better in the figure - with better lines perhaps

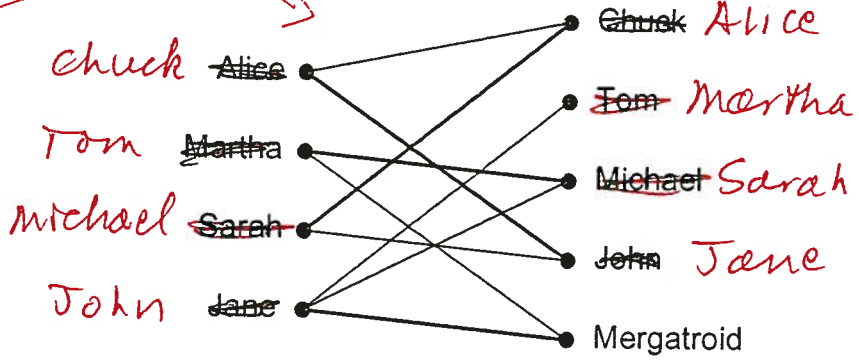


Figure 5K: One possible matching for the men.

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.

A famous result known as

Subsubsection (no #)

9.7.3 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:

man-likes-woman women men women men women Tom John martha Sarah men

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 9.7.3. *A matching for a set of girls with a set of boys 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 n, 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 $Y \subseteq B$.[1] 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

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

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.

9.7.4 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

A matching is said to be perfect if every node in the graph is incident to an edge in the matching.

M-10

has an edge of the matching incident to it. In any graph, the set $N(S)$, of *neighbors* of some set, S, of vertices is the set of all vertices adjacent to some vertex in S. That is,

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

S is called a *bottleneck* if

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

Theorem 9.7.4 (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

← subsubsection (no #)

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

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

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 9.7.5. 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 9.7.6. *Every degree-constrained bipartite graph satisfies the matching condition for L.*

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 ∎

INSERT M

Corollary 5E

For example, the graph in Figure 5 is degree constrained since every node on the left is adjacent to at least two nodes on the right while every node on the right is incident to at most two nodes on the left.

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

Definition 5P: A graph is said to be regular if every node has the same degree.

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

Lemma 9.76: Let G be a bipartite graph with vertex partition L, R ~~If G is degree~~ where $|L| \leq |R|$. If G is degree-constrained, then no subset of L is a bottleneck.

Corollary 5L: Let G be a bipartite graph with vertex partition L, R where $|L| \leq |R|$. If G is degree-constrained, then there is a matching that covers L.

Proof: Combine Lemma 9.7.6 with Theorem 9.7.4. ∎

we can now prove:

Theorem 5M: Every regular bipartite graph has a perfect matching.

Proof: Let G be a regular bipartite graph with vertex partition L, R where $|L| \leq |R|$. Since G is regular, we know by Corollary 5L that there must be a matching in G that covers L. Since G is regular, we also know that $|L| = |R|$ and thus the matching must also cover R. This means that every node in G is incident to an edge in the matching and thus G has a perfect matching. ∎

~~Corollary 5M will prove to be very useful in Chapter 6 where we need to assert that certain graphs have perfect matchings~~

~~Corollary 5M.1 is a~~ H-15

Theorem 5M is ~~a~~ a surprisingly useful tool in graph theory. For example, we will use it in chapter 6 when we are figuring out how to route data paths in a Benes network.

H-16

Everything is crossed out on this page.

bottleneck. So there are no bottlenecks, proving that the degree-constrained graph satisifies the matching condition.

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.

9.7.5 Problems

Class Problems

Exam Problems

Homework Problems

Corollary 5L: Let $G = (V, E)$ be a degree constrained bipartite graph. Then there is a matching for G that covers L.

Corollary 5M: There is a perfect matching in any regular bipartite graph.

Proof: If every node has the same degree, then $|L| = |R|$ and a matching that covers L also covers R. Such a matching must exist by Lemma 9.7.6. ∎

Definition 5X: A matching is said to be perfect if every node is incident to an edge in the

THIS IS THE END OF INSERT H.