Stable Marriages; Simple Graphs

1 The Stable Marriage Problem

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

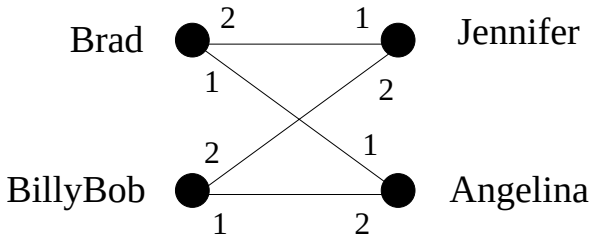
1.1 The Problem

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 he has a complete list of all the girls ranked according to his preferences, with no ties. Likewise, each girl has her rank list of all of the boys.

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 off boys and girls: every boy must marry exactly one girl and vice-versa —no polygamy. 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.

Here's the difficulty: suppose *every* boy likes Angelina best, and every girl 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 doing 6.042 homework 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.

Copyright © 2007, Prof. Albert R. Meyer. All rights reserved.

Definition 1.1. A matching is *stable* iff it has 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.

It is something of a surprise that there always is a stable matching among a group of boys and girls, but there is, 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 1 shows a situation with a love triangle and a fourth person who is everyone's last choice. In this figure Mergatoid's preferences aren't shown because they don't even matter.

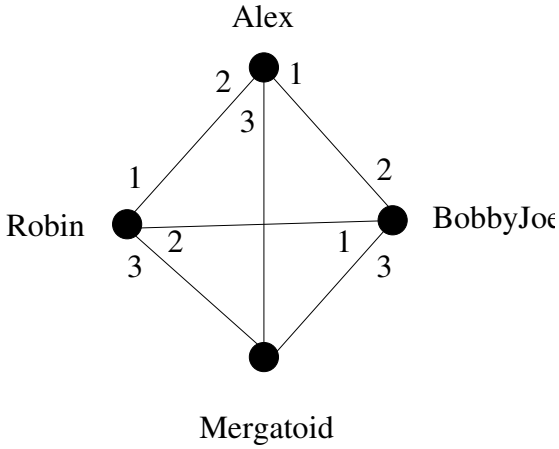


Figure 1: Some preferences with no stable buddy matching.

Let's see why there is no stable matching:

Lemma. *There is no stable buddy matching among the four people in Figure 1.*

Proof. We'll prove this by contradiction.

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

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

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

1.2 The Mating Ritual

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

Morning: Each girl stands on her balcony. Each boy stands under the balcony of his favorite among the girls on his list, and he serenades her. If a boy has no girls left on his list, he stays home and does his 6.042 homework.

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

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

Termination condition: When every girl has at most one suitor, the ritual ends with each girl marrying her suitor, if she has one.

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

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

1.3 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 1.2. 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 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 mathematicallly 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.

1.4 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.

1.5 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:

> For every girl, G, and every boy, B, if G is crossed off B's list, then G has a suitor whom she prefers over B.

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

Notice that P also holds on the first day, since every girl 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.

Theorem 1.3. *Everyone is married by the Mating Ritual.*

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 girl has a favorite boy 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 1.4. *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: the claim also holds in this case. □

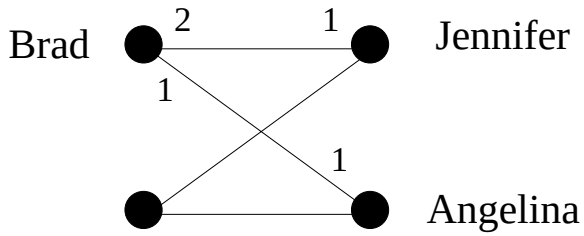
1.6 ...Especially the Boys

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 1.5. 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 1.6. *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, call 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 hasn't crossed off his optimal girl. So

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

By the definition of an optimal girl, there must be some stable set of marriages in which Keith gets his optimal girl, 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 1.7. *The Mating Ritual marries every girl to her pessimal spouse.*

Proof. Say Nicole and Keith marry each other as a result of the Mating Ritual. By the previous Theorem 1.6, Nicole is Keith's optimal spouse, and so in any stable set of marriages,

> 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

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. □

1.7 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 unbeknownst to 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.) 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.[1]

MIT Math Prof. Tom Leighton, who regularly teaches 6.042 and also founded the internet infrastructure company, Akamai, reports another application. Akamai 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 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. The servers have preferences based on latency and packet loss; the traffic has preferences based on the cost of bandwidth.

2 Simple Graphs

2.1 Introduction

Graphs are an incredibly useful structure in Computer Science! They arise 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.

[1] 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.

But first we are going to talk about something else. Namely, sex. The question that we'll address is, on average, who has more opposite-gender partners, men or women? This has been the subject of many studies. In one of the largest, researchers from the University of Chicago interviewed a "random sample" of 2500 people over several years to try to get an answer to this question. Their study, published in 1994, and entitled *The Social Organization of Sexuality* found that on average men have 74% more opposite-gender partners than women, which fits right in with stereotype of male promiscuity versus female sexual selectivity.

Other studies have found that the disparity is even larger. In particular, ABC News claims 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, claimed to be one of the most scientific ever done with only a 2.5% margin of error. It was called "American Sex Survey: A peak between the sheets"—- hmmmmmm, doesn't sound so scientific. 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.

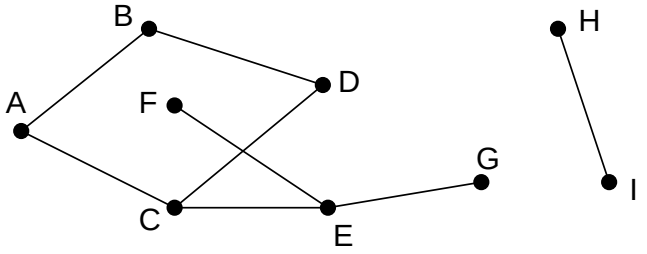
Probably that last part about going where few scientific surveys have gone before is pretty accurate!

And 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.

2.2 Definition of Simple Graph

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



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

Definition 2.1. 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 in dots-and-lines diagram above is a captured by the following simple graph:

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

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.

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. 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.

2.2.1 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*, and adjacent vertices are sometimes called *neighbors*. 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 2.1 are worth noting right from the start:

1. Simple graphs do not have edges going from a vertex back around to itself (called a *self-loop*).
2. There is at most one edge between two vertices of a simple graph.
3. Simple graphs have at least one vertex, though they might not have any edges.

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

For the rest of these Notes and much of next week's Notes, we'll only be considering simple graphs, so we'll just call them "graphs" from now on.

2.3 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 men, and W, which contains all the women.[2] We'll put an edge

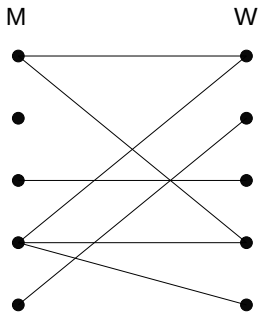


Figure 2: The sex partners graph

between a man and a woman iff they have been sexual partners. This graph is pictured in Figure 2 with men on the left and women on the right.

Actually, this is a pretty hard graph to figure out, let alone draw. The graph is *enormous*: the US population is about 300 million, so $|V| \approx 300M$! Of these, approximately 50.8% are female and 49.2% are male, so $|M| \approx 147.6M$, and $|W| \approx 152.4M$. And we don't even have trustworthy estimates of how many edges there are, let alone exactly which couples are adjacent.

But it turns out that we don't need to know any of this —we just need to figure out the relationship between the average number of partners per male and partners per female. To do this, we note that every edge is incident to exactly one M vertex (remember, we're only considering male-female relationships); so the sum of the degrees of the M vertices equals the number of edges. For the same reason, the sum of the degrees of the F vertices equals the number of edges. So these sums are equal:

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

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

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

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

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

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

Now the Census Bureau reports that there are slightly more females than males in America; in particular $|F| / |M|$ is about 1.035. So we know that on average, males have 3.5% more opposite-gender partners than females, and this tells us nothing about any sex's promiscuity or selectivity. Rather, it just has to do with the relative number of males and females. Collectively, males and females have the same number of opposite gender partners, since it takes one of each set for every

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

partnership, but there are fewer men, so they have a higher ratio. So 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 men exaggerate their number of partners —or maybe women downplay theirs —but these explanations are speculative. Interestingly, the principal author of the government study reported that she knew the results had to be wrong, but felt she had no choice but to publish the data collected in the survey.

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. Well, that just follows from the definition of "minority"!

2.4 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:

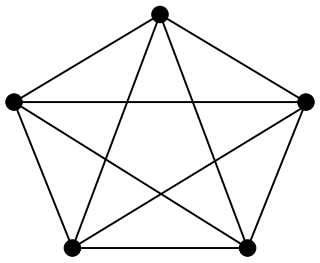
Theorem 2.2. *The sum of the degrees of the vertices in a graph equals twice the number of edges.*

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

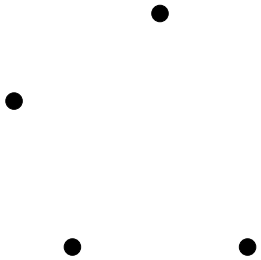
Theorem 2.2 is sometimes called the *Handshake Theorem*: if we total up the number of people each person at a party shakes hands with, the total will be twice the number of handshakes that occurred.

2.5 Some Common Graphs

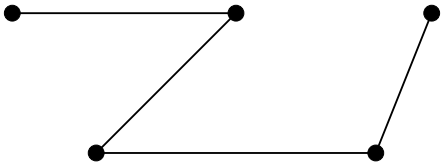
Some graphs come up so frequently that they have names. The *complete graph* on n vertices, also called K_n, has an edge between every two vertices. Here is K_5:



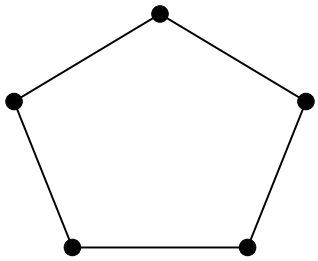
The *empty graph* has no edges at all. Here is the empty graph on 5 vertices:



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

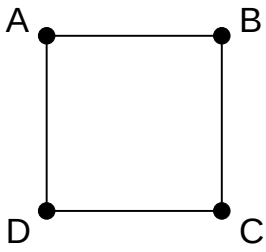


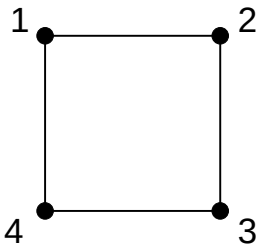
And here is C_5, a *simple cycle* with 5 vertices:



2.6 Isomorphism

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





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

Fortunately, we can neatly capture the idea of "looks the same." 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. For example, take the following bijection between vertices in the two graphs above:

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

Now 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. Therefore, the two graphs are isomorphic. The bijection itself is called an *isomorphism*.

Definition 2.3. If G_1 is a graph with vertices, V_1, and edges, E_1, and likewise for G_2, then G_1 is *isomorphic* to G_2 iff there exists a **bijection**, $f : V_1 \to V_2$, such that for every pair of vertices $u, v \in V_1$:

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

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

Two isomorphic graphs may be drawn very differently. For example, here are two different ways of drawing C_5:

Isomorphism captures all 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. So a property like "having three vertices of degree 4" is preserved under isomorphism, while "having a vertex that is an integer" is not preserved. In particular, if one graph has three vertices of degree 4 and another does not, they can't be isomorphic. Similarly, if one graph has an edge that is incident to a degree 8 vertex and a degree 3 vertex, then any isomorphic graph must also have such an edge.

Looking for properties like these 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 which 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.

3 Connectedness

3.1 Paths and Simple Cycles

A *path* in a graph describes how to get from one vertex to another following edges of the graph. Formally,

Definition 3.1. A ***path*** in a graph, G, is a sequence of $k \geq 0$ vertices

$$v_0, \ldots, v_k$$

such that v_i—v_{i+1} is an edge of G for all i where $0 \leq i < k$. The path is said to ***start*** at v_0, to ***end*** at v_k, and ***length*** of the path is defined to be k. An edge, e, is ***traversed*** n ***times*** by the path if there are n different values of i such that edge v_i—v_{i+1} is e.

The path is ***simple***[3] iff all the v_i's are different, that is, $v_i = v_j$ only if $i = j$.

For example, the graph in Figure 3 has a length 6 simple path A,B,C,D,E,F,G. This is the longest simple path in the graph.

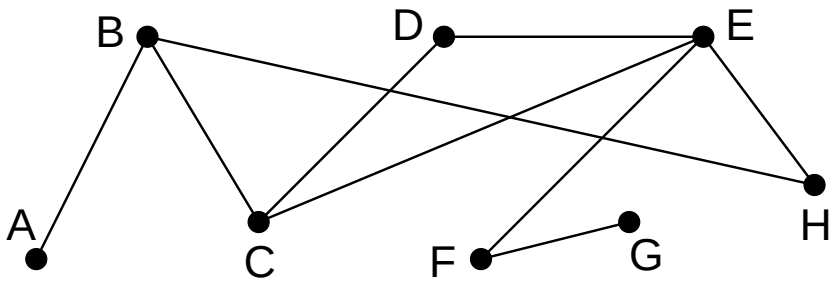


Figure 3: *A graph with 3 simple cycles.*

Notice that 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. The length 6 path A,B,C,D,E,F,G is actually a sequence of seven vertices.

A *cycle* can 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 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.)

[3]Heads up if you read another graph theory text: what amounts to paths are commonly referred to as "walks," and simple paths are referred to as a just "paths". Likewise, what we will call *cycles* and a *simple cycles* are commonly called "closed walks" and a just "cycles".

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 *positive length* cycle that doesn't cross or backtrack on itself. For example, the graph in Figure 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 positive length path 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]

3.2 Connected Components

Definition 3.2. Two vertices in a graph are said to be ***connected*** if there is a path that begins at one and ends at the other.

By convention, every vertex is considered to be connected to itself by a path of length zero.

The diagram in Figure 4 looks like a picture of three graphs, but is intended to be a picture of *one* graph. This graph consists of three pieces (subgraphs). Each piece by itself is connected, but there are no paths between vertices in different pieces.

Figure 4: *A graph with 3 connected components.*

Definition 3.3. A graph is said to be ***connected*** if every pair of vertices are connected.

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

[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 cyle is which paths describe it.

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. This leads to the following definition:

Definition 3.4. Two vertices in a graph are k-connected if they remain connected in any subgraph obtained by deleting $k-1$ edges. A graph with at least two vertices is k-connected[5] if every pair of its vertices are k-connected.

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

For example, in the graph in Figure 3, vertices B and E are 2-connected, G and E are 1-connected, and no vertices are 3-connected. The graph as a whole is only 1-connected.

More generally, any simple cycle is 2-connected, and the complete graph, K_n, is $(n-1)$-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-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-connected, then there are k edge-disjoint paths connecting them. It takes some ingenuity to prove this even for the case $k = 2$.

3.4 Connection by Simple Path

Where there's a path, there's a simple path. This is sort of obvious, but proving it carefully provides a nice illustration of the Well-ordering Principle.

Lemma 3.5. *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. □

[5] The literature commonly calls this "k-edge-connected", reserving "k-connected" for a similar concept based on the removal of vertices.

Actually, we proved something stronger:

Corollary 3.6. *For any path of length k in a graph, there is a simple path of length* at most *k with the same endpoints.*

3.5 The Minimum Number of Edges in a Connected Graph

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

Theorem 3.7. *Every graph with v vertices and e edges has at least $v - e$ connected components.*

Of course for Theorem 3.7 to be of any use, there must be fewer edges than vertices.

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

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

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

Inductive step: Now we assume that the induction hypothesis holds for every e-edge graph in order to prove that it holds for every $(e + 1)$-edge graph, where $e \geq 0$.

Consider a graph, G, with $e+1$ edges and k vertices. We want to prove that G has at least $v-(e+1)$ connected components.

To do this, remove an arbitrary edge a—b and call the resulting graph G'. By the induction assumption, G' has at least $v - e$ connected components.

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

The theorem now follows by induction. □

Corollary 3.8. *Every connected graph with e vertices has at least $e - 1$ edges.*

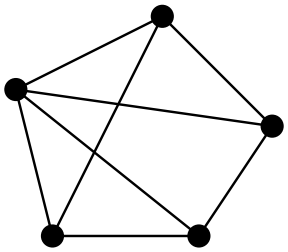
A couple of points about the proof of Theorem 3.7 are worth noting. First, notice that we used induction on the number of edges in the graph. This is very common in proofs involving graphs, and so is induction on the number of vertices. When you're presented with a graph problem, these two approaches 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 e-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. You'll see an example in recitation. Always use shrink-down, grow-back arguments, and you'll never fall into this trap.

4 Coloring Graphs

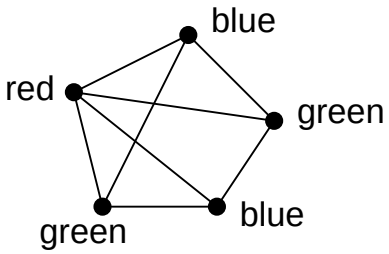
In the "Sex in America" graph In Week 5 Notes, we used edges to indicate an affinity between two nodes, but having an edge represent a *conflict* between two nodes also turns out to be really useful. For example, each term the MIT Schedules Office must assign a time slot for each final exam. This is not easy, because some students are taking several classes with finals, and a student can take only one test during a particular time slot. The Schedules Office wants to avoid all conflicts. Of course, you can make such a schedule by having every exam in a different slot, but then you would need hundreds of slots for the hundreds of courses, and exam period would run all year! So, the Schedules Office would also like to keep exam period short.

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



6.002 and 6.042 cannot have an exam at the same time since there are students in both courses, so there is an edge between their nodes. On the other hand, 6.042 and 6.170 can have an exam at the same time if they're taught at the same time (which they sometimes are), since no student can be enrolled in both (that is, no student *should* be enrolled in both when they have a timing conflict). Next, identify each time slot with a color. For example, Monday morning is red, Monday afternoon is blue, Tuesday morning is green, etc.

Assigning an exam to a time slot is now equivalent to coloring the corresponding vertex. The main constraint is that *adjacent vertices must get different colors* —otherwise, some student has two exams at the same time. Furthermore, in order to keep the exam period short, we should try to color all the vertices using as *few different colors as possible*. For our example graph, three colors suffice:



This coloring corresponds to giving one final on Monday morning (white), 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. The minimum value of k for which a coloring exists is called the ***chromatic number***, $\chi(G)$, of 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).

4.1 Degree-bounded Coloring

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

Theorem 4.1. *A graph with maximum degree at most k is $(k+1)$-colorable.*

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

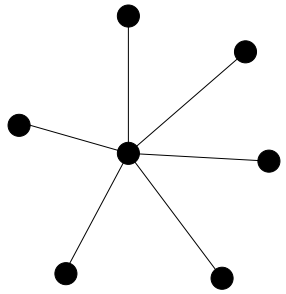
Proof. We use induction on the number of vertices in the graph, which we denote by n. Let $P(n)$ be the proposition that an n-vertex graph with maximum degree at most k is $(k+1)$-colorable.

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

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

Sometimes $k+1$ colors is the best you can do. For example, in the complete graph, K_n, every one of its n vertices is adjacent to all the others, so all n must be assigned different colors. Of course n colors is also enough, so $\chi(K_n) = n$. So K_{k+1} is an example where Theorem 4.1 gives the best possible bound. This means that Theorem 4.1 also gives the best possible bound for *any* graph with degree bounded by k that has K_{k+1} as a subgraph.

But sometimes $k+1$ colors is far from the best that you can do. Here's an example of an n-node star graph for $n = 7$:



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

4.2 Why coloring?

Coloring problems come up in all sorts of applications. For example, at Akamai, a new version of software is deployed over each of 20,000 servers every few days. The updates cannot be done at the same time since the servers need to be taken down in order to deploy the software. Also, the servers cannot be handled one at a time, since it would take forever to update them all (each one takes about an hour). Moreover, certain pairs of servers cannot be taken down at the same time since they have common critical functions. This problem was eventually solved by making a 20,000 node conflict graph and coloring it with 8 colors – so only 8 waves of install are needed!

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

Coloring also comes up 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 mentioned in Week 1 Notes. The question is how many colors are needed to color a map so that adjacent territories get different colors? This is the same as the number of colors needed to color a graph that can be drawn in the plane without edges crossing. A proof that four colors are enough for the *planar* graphs was acclaimed when it was discovered about thirty years ago. Implicit in that proof was a 4-coloring procedure that takes time proportional to the number of vertices in the graph (countries in the map). On the other hand, it's another of those million dollar prize questions to find an efficient procedure to tell if a planar graph really *needs* four colors or if three will actually do the job. But it's always easy to tell if an *arbitrary* graph is 2-colorable, as we show next week.

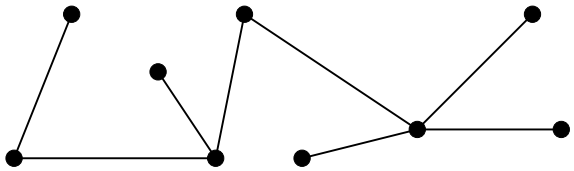
5 Trees

Trees are a fundamental data structure in Computer Science, and there are many kinds, for example rooted, ordered, or binary trees. In this section we focus on the purest kind of tree. Namely, we use the ***tree*** to mean a connected graph without simple cycles.

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

5.1 Tree Properties

Here is an example of a tree:



A vertex of degree one is called a ***leaf***. In this example, there are 5 leaves.

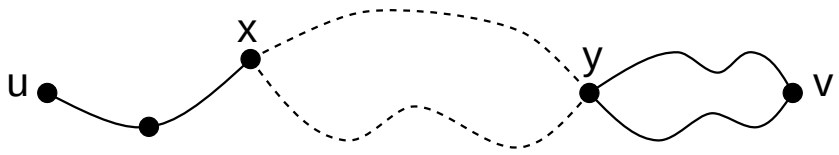
The graph shown above would no longer be a tree if any edge were removed, because it would no longer be connected. The graph would also not remain a tree if any edge were added between two of its vertices, because then it would contain a simple cycle. Furthermore, note that there is a unique path between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

Theorem 5.1. *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 a 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 $0 + 1 = 1$.

   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.

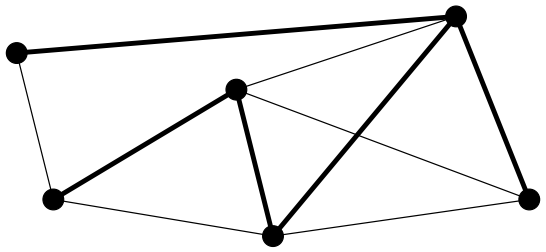
   We will let the reader verify that deleting a vertex of degree 1 (and its incident edge) from any connected graph leaves a connected subgraph. So by (1), deleting v and its incident edge gives a smaller tree, and this smaller tree has one more vertex than edge by induction. If we reattach the vertex, 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.

5.2 Spanning Trees

Trees are everywhere. In fact, every connected graph contains a 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 5.2. *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:

$$v_0\text{—}v_1, v_1\text{—}v_2, \ldots, v_n\text{—}v_0$$

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. □