Massachusetts Institute of Technology
6.042J/18.062J: Mathematics for Computer Science
Professor Albert R. Meyer

Handout 24
March 9, 1998

Notes on Graphs

1 Graph Definitions

Graphs are mathematical objects used heavily in Computer Science. Often one can reduce a real-world problem to a purely mathematical statement about graphs. If the graph problem can be solved, then — in principle at least — the real-world problem is solved.

A nuisance in first learning graph theory is that there are so many definitions. They all correspond to intuitive ideas, but can take a while to absorb. Worse, the same thing often has several names and can even have several equivalent definitions!

1.1 Simple Graphs

A *simple graph* is a pair of sets (V, E). Elements of V are called *vertices*. An element of E is called an *edge*. The basic property of an edge in a simple graph is that it adjoins two vertices. Formally, we identify an edge with the two vertices it adjoins. That is, the elements of E are specified to be subsets of V of size two.

Graphs are also sometimes called *networks*. Vertices are also sometimes called *nodes*. Edges are sometimes called *arcs*. If $u \neq v$ are vertices of a simple graph and the set $\{u, v\}$ is an edge of the graph, this edge is said to be *incident* to u and v. Equivalently, u and v are said to be *adjacent* or *neighbors*. Phrases like, "an edge joins u and v" and "the edge between u and v" are common.

Graphs can be nicely represented with a diagram of dots for vertices and lines for edges as shown in Figure 1.

1.2 Not Simple Graphs

Simple graphs are just one kind of graph; there are other kinds, often not as simple.

Multigraphs

In a simple graph, there are either zero or one edges joining a pair of vertices. In a *multigraph*, multiple edges are permitted between the same pair of vertices. There may also be edges that connect a vertex to itself; these are called *self-loops*[1]. Figure 2 depicts a multigraph with self-loops.

[1]This requires that an edge be something more than just a set of two vertices — if that's all it was, it would be the only possible edge between the two vertices. One way to formalize this is to say that the edges of a multigraph are a set E — whose elements may be anything at all — along with a function from E into the subsets of V of size one or two.

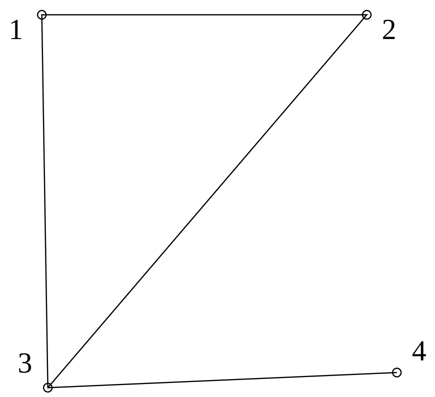


Figure 1: *This is a picture of a graph $G = (V, E)$. There are 4 vertices and 4 edges. The set of vertices V is $\{1, 2, 3, 4\}$. The set, E, of edges is $\{\{1, 2\}, \{1, 3\}, \{2, 3\}, \{3, 4\}\}$. Vertex 1 is adjacent to vertex 2, but is not adjacent to vertex 4.*

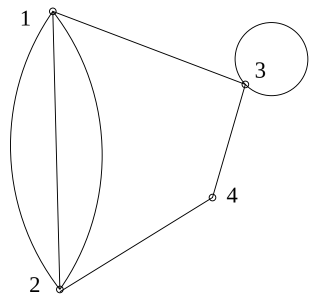


Figure 2: *This is a picture of a multigraph with a self-loop. In particular, there are three edges connecting vertices 1 and 2 and there is a self-loop on vertex 3.*

Directed Graphs

Like a simple graph, a *directed graph* or *digraph* is a pair of sets (V, E) where V is a set whose elements are called vertices. Now the edges are regarded not as lines, but as arrows going from a *start* (or *tail*) vertex to an *end* (or *head*) vertex. Formally, an edge in E is specified to be an ordered pair of vertices. In other words, $E \subseteq V \times V$, where $V \times V$, the Cartesian product of V with itself, is the set of all ordered pairs of elements of V. Figure 3 depicts a directed graph.

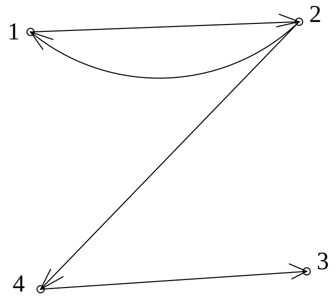


Figure 3: *This is a picture of a directed graph or digraph.*

Weighted Graphs

Sometimes it is useful to associate a number, often called its *weight*, with each edge in a graph. Such graphs are called *edge-weighted* or simply *weighted* graphs; they may be simple, directed, multi, etc. The weight of an edge (u, v) of a digraph is often denoted $w(u, v)$. More generally, edges (or nodes) may be labelled with elements from some designated set of labels — these are called edge (or node) *labelled* digraphs, simple graphs, multigraphs, etc.

2 Graphs in the Real World

There are many real-world phenomena that can be described nicely by graphs. Here are some examples:

Computer network The set of vertices V represents the set of computers in the network. There is an edge (u, v) iff there is a direct communication link between the computers corresponding to u and v.

Airline Connections Here the vertices are airports and edges are flight paths. We could indicate the direction that planes fly along each flight path by using a directed graph. We could use weights to convey even more information. For example, $w(i, j)$ might be the distance between airports i and j, or the flying time between them, or even the air fare. The edges might also be labelled with the set of call signs (`TW`, `AA`, ...) of airlines using that flight path.

Precedence Constraints Suppose you have a set of jobs to complete, but some must be completed before others are begun. (For example, Atilla advises you always pillage *before* you burn.) Here the vertices are jobs to be done. Directed edges indicate constraints; there is a directed edge from job u to job v if job u must be done before job v is begun.

Program Flowchart Each vertex represents a step of computation. Directed edges between vertices indicate control flow.

Two-Player Game Tree All of the possibilities in a board game like chess can be represented in a graph. Each vertex stands for one possible board position. (For chess, this is a very big graph!)

Some graphs appear so frequently that they have names. The most important examples are shown in Figure 4.

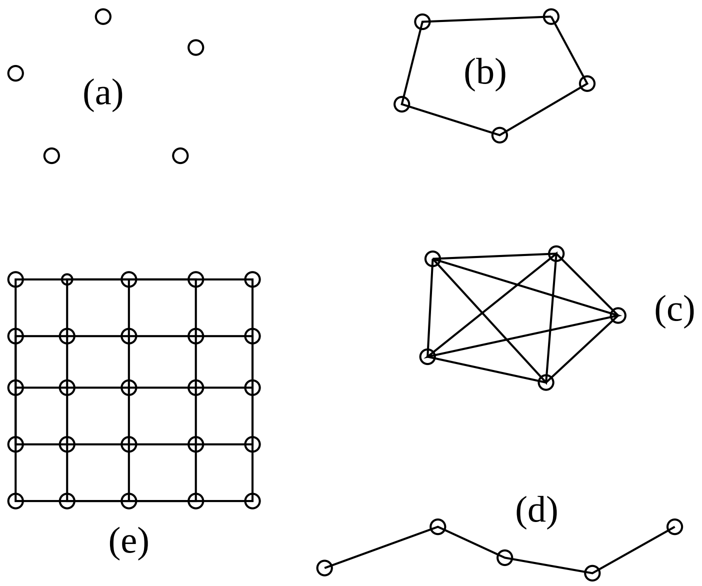


Figure 4: *The types of graph shown here are so common that they have names. (a) The empty graph or* anticlique *on five vertices, A_5. An empty graph has no edges at all. (b) The cycle on five vertices, C_5. (c) The complete graph on five vertices, K_5. A complete graph has an edge between every pair of vertices. (d) A five-vertex line graph. (e) A 5×5 2-dimensional mesh.*

3 Graph Isomorphism

Graphs are intended to be abstract data types. This means what matters about a graph is only its "shape": which vertices are incident to which edges, but not what the vertices actually are. For example the simple graph whose vertices are the integers $1, \ldots, 2n$ with an edge between two vertices iff they have the same parity (that is, both are even or both are odd), has the "same shape" as the graph whose vertices are $-1, -2, \ldots, -2n$ with an edge between two vertices iff either both vertices are $< -n$ or both are $\geq -n$, since in each case, the graph has two disjoint sets of n vertices, with all possible edges between vertices in the same set and none between a vertex and any of the vertices in the opposite set. Technically we say the graphs are *isomorphic* when their vertices can be put in exact correspondence so that an edge between two vertices in one graph corresponds exactly to an edge between corresponding vertices in the other graph.

It helps to say this in precise mathematical style. Namely, digraphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are isomorphic iff there is a bijection $f : V_1 \to V_2$ such that for all $u, v \in V_1$

$$(u, v) \in E_1 \iff (f(u), f(v)) \in E_2.$$

The bijection, f, is called an *isomorphism* between the graphs. For example, the function $f : \{-1, \ldots, -2n\} \to \{1, \ldots, 2n\}$, where $f(k) = -2k$ if $-n \leq k \leq -1$ and $f(k) = 2(2n + k) + 1$ if $-2n \leq k < -n$, is an isomorphism between the two graphs described above.

4 Properties of Graphs

When one is confronted by a new graph — say in a dark alley — there is often a need to study properties at a higher-level than, "Is vertex u connected to vertex v?" Some questions that might arise are, "How many edges are incident to each vertex?", "Is the graph all in one piece or in several pieces?", "How can we color the vertices?" (The last may seem odd, but comes up surprisingly often; we'll see why in a minute.) This section discusses some of these higher-level properties of graphs.

4.1 Vertex Degree

The *degree* of a vertex is the number of edges incident to it. The degree of vertex v is often denoted $d(v)$. In a digraph, we might also examine the *in-degree* (resp. *out-degree*) of a vertex, namely the number of edges into (resp. out of) a vertex.

For example, referring to Figure 4, every vertex in an empty graph has degree 0, but every vertex in a cycle has degree 2. A simple example of what we mean by a "higher-level" property of simple graphs is:

Theorem 4.1 *The sum of the degrees of the vertices in a simple graph equals twice the number of edges.*

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

Exercise:
(a) The previous proof is not as precise as we desire at the beginning of 6.042. Rewrite the proof more carefully as an induction on the number of edges in a simple graph.
(b) Extend Theorem 4.1 to multigraphs.
(c) Extend Theorem 4.1 to digraphs.

Here is a puzzle that can be addressed with graphs. There is a party. Some people shake hands an even number of times and some shake an odd number of times. Show that an even number of people shake hands an odd number of times.

We can represent the party by a graph. (Yeah, right.) Each person is represented by a vertex. If two people shake hands, then there is an edge between the corresponding vertices. This reduces the problem to the following theorem:

Theorem 4.2 *In every graph, there are an even number of vertices of odd degree.*

Proof. Partitioning the vertices into those of even degree and those of odd degree, we know

$$\sum_{v \in V} d(v) = \sum_{d(v) \text{ is odd}} d(v) + \sum_{d(v) \text{ is even}} d(v)$$

The value of the lefthand side of this equation is even, and the second summand on the righthand side is even since it is entirely a sum of even values. So the first summand on the righthand side must also be even. But since it is entirely a sum of odd values, it must must contain an even number of terms. That is, there must be an even number of vertices with odd degree. ∎

Two graphs with the same shape will of course have the same pattern of vertex degrees – that's one of things that "same shape" should imply. More precisely, if f is an isomorphism between two graphs, then it is easy to show that v and $f(v)$ have the same degree for any vertex v. We say that the degree of a vertex is *invariant* under graph isomorphism. Since f is a bijection, it follows that isomorphic graphs must have exactly the same numbers of vertices of any degree d. That is, the number of vertices of degree d in a graph is an invariant under isomorphism. Isomorphic graphs must also have the same number of pairs of vertices of degree d_1 and degree d_2 which are adjacent – another invariant under isomorphism. If two graphs are found to have different values of some invariant, then they are cannot be isomorphic. Finding such invariants can be a simple means to prove that two perhaps roughly similar looking graphs, are not actually isomorphic (see Rosen, 7.3, Examples 9, 10).

4.2 Chromatic Number

Time to discuss final exams. The MIT Schedules Office needs to assign a time slot for each final. 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, but wants to make the exam period as short as possible.

This scheduling problem can be represented by a graph. Let each vertex represent a course. Put an edge between two vertices if there is some student taking both courses. Identify each possible time slot with a color. For example, Monday 9–12 is red, Monday 1–4 is blue, Tuesday 9–12 is green, etc.

If there is an edge between two vertices with the same color, then a conflict exam will have to be scheduled because there is a student who has to take exams for the courses represented by the vertices, but the exams are scheduled at the same time. Everyone wants to avoid conflict exams, so the registrar would like to color each vertex of the graph so that no adjacent vertices have the same color; to keep exam period as short as possible, the registrar would like to use the minimum possible number of colors. An example is shown in Figure 5.

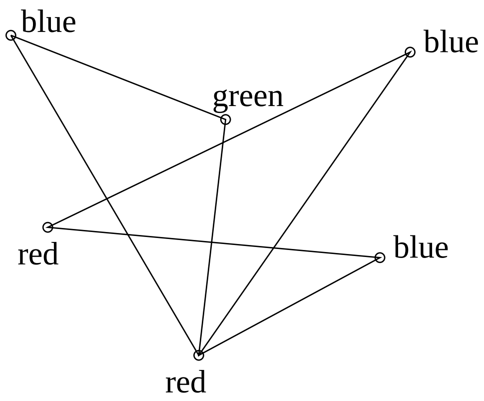


Figure 5: *This graph represents the exam scheduling problem. Each vertex stands for a course. An edge between two vertices indicates that a student is taking both courses and therefore the exams cannot be scheduled at the same time. Each exam time slot is associated with a color. A schedule that creates no conflicts corresponds to a coloring of the vertices such that no adjacent vertices receive the same color.*

In general, the minimum number of colors needed to color the vertices of a graph G so that no two adjacent vertices are the same is called the *chromatic number* and is written $\chi(G)$. For example, if G is the 3-cycle or triangle graph, then $\chi(G) = 3$.[2]

The only completely general way known for finding the chromatic number of a graph is to exhaustively try all possible colorings; the exhaustive approach really is exhausting, and it becomes prohibitive even for graphs with only a few dozen vertices. Sometimes we can take advantage of the structure of special classes of graphs to get a better grip on their colorings. Also, we can at least put some general upper bounds on the chromatic number. For example, the chromatic number of an n-vertex graph is certainly at most n, since every vertex could be assigned a different color.

Suppose that we tried to do better by coloring vertices in an arbitrary order, but using a new color only when forced to do so. This strategy lies behind the following recursive algorithm. The input is a graph, and the output is a graph with appropriately colored vertices.

- Pick a vertex v. Remove v and all incident edges from the graph.
- If the graph is not empty, color the remainder recursively.
- Add v back to the graph. Color v differently from all neighbors, using a new color only if necessary.

Theorem 4.3 *The preceding algorithm colors a graph with at most $p + 1$ colors, where p is the maximum degree of any vertex.*

This theorem implies, for example, that a graph with thousands of vertices, each of degree 3, requires at most 4 colors. The proof is surprisingly easy:

[2] χ is the Greek letter "chi" — pronounced "key" by the ancient Greeks and "kye" by mathematicians and frats.

Proof. The proof is by induction. Let $P(n)$ be the predicate that the preceding algorithm colors every graph in which every vertex has degree at most p using at most $p+1$ colors.

In the base case, $P(1)$, there is a single vertex with degree zero. In this case, $p = 0$ and the algorithm requires $p + 1 = 1$ colors.

In the inductive step, assume $P(n)$ to prove $P(n+1)$. Let G' be the graph obtained from G by removing vertex v and incident edges. No vertex in G' has degree greater than p, since removing a vertex and incident edges can not increase the degree of any other vertex. (This is what we had to check to avoid "buildup error" — see below.) By induction, the algorithm (applied recursively) colors G' with at most $p+1$ colors. Now we add back vertex v. Since v has at most p neighbors and there are $p+1$ colors available, there is always one color left over for vertex v. Therefore, the algorithm colors G with at most $p+1$ colors. ■

4.3 Paths in Graphs

Is a graph all in one piece or composed of several pieces? To walk from one vertex to another, how many edges must one cross? Are there many different routes or just one? These are questions about *connectivity.*

A *path* in a digraph from a vertex u to a vertex v is a sequence of $k \geq 1$ edges

$$(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$$

such that $v_0 = u$ and $v_k = v$. A path may contain the same edge multiple times. The *length* of the path is k. The sequence of *sequence of vertices on the path* is the sequence $v_0, v_1, v_2, \ldots, v_k$. Vertex u is said to be *connected* to vertex v iff $u = v$ or there is a path from u to v. The definitions for simple graphs are the same except that the ordered pair (v_i, v_{i+1}) is replaced by $\{v_i, v_{i+1}\}$. A path is *simple* when no vertex occurs more than once in the sequence of vertices on the path.

Lemma 4.4 *If vertex u is connected to vertex $v \neq u$ in a graph, then there is a simple path from u to v.*

The proof provides a nice example illustrating the Well-ordering Principle.

Proof. Since $u \neq v$, there is a path from u to v. By the Well-ordering Principle, there must be a minimum length path

$$(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$$

from u to v. We claim this path must be simple. This is clearly the case if $k = 1$, since then the path consists of a single edge from u to v. Given that $k > 1$, we prove the claim by contradiction.

Namely, assume that some vertex on the path occurs twice. More precisely, there are $i, j \in \mathbb{N}$ such that $i < j \leq k$ with $v_i = v_j$. Then removing the sequence of $j - i > 0$ edges

$$(v_i, v_{i+1}), \ldots, (v_{j-1}, v_j)$$

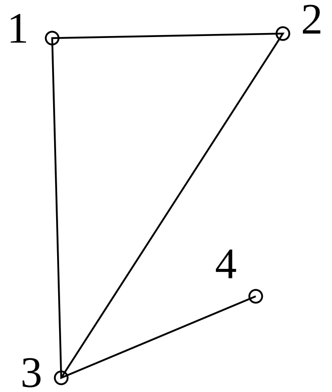
from the path yields a path from u to v of length $< k$, contradicting the fact that k is minimal. ■

A *cycle* (also called a *circuit*) in a graph is a path from a vertex to itself, i.e. a sequence of edges $(u, v_1), \ldots, (v_{k-1}, u)$. A *simple cycle* is a cycle in which only the first and last vertex occurring on the path are the same and no edge occurs twice[3]. Another way to say this is that a simple cycle is either a self-loop or is a cycle which becomes a simple path when its first or last edge is deleted.

4.4 Adjacency Matrices

We defined a graph in terms of a set of vertices and a set of edges. A graph can also be described by a matrix.

Let $v_1, \ldots, v_n$ be list of the vertices of a digraph, G. The *adjacency matrix* of G is an $n \times n$ matrix of zeroes and ones. The entry in row i and column j of the matrix is one iff there is an edge from v_i to vertex v_j. An example is shown in Figure 6.



$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Figure 6: *This is a graph with list of vertices (1, 2, 3, 4), and its associated* 4×4 *adjacency matrix. The entry in row i and column j of the matrix is a 1 if there is an edge* (i, j) *in the graph. All other entries are 0.*

Weighted or labelled digraphs can also be described well by matrices: let the label of edge (v_i, v_j) be the matrix entry in row i, column j.

Observe that in a digraph with n vertices, the paths from a vertex, u, to a vertex, v, can be partitioned (divided into nonoverlapping groups) according to their next-to-last vertex. That is, the paths from u to v can be partitioned into at most n sets, one for each vertex adjacent to v. The kth set consists of those paths, if any, which begin with a path from u to v_k followed by an edge from v_k to v. Matrix multiplication can now be given a neat graphical interpretation.

Theorem 4.5 *Let A be the $n \times n$ adjacency matrix of a digraph G, and let A^m be its mth power for $m > 0$. Then A^m_{ij}, the ijth entry of A^m, is exactly equal to the number of distinct paths of length m from vertex i to vertex j.*

[3] The condition that no edge occurs twice is only needed for undirected graphs, where we don't want going back and forth on the same edge to count as a simple cycle.

Proof. By induction on k.

Base case $k = 1$ holds by definition.

To prove the induction case, note that from the partitioning of paths by next-to-last vertices described above, the number of paths from v_i to v_j of length $m > 1$ is the sum of the number of paths of length $m - 1$ from v_i to v_k with the sum taken over all k such that there is an edge from v_k to v_j. This is the same as

$$\sum_{k=1}^{n} |\{\text{length } m-1 \text{ paths from } v_i \text{ to } v_k\}|\, A_{kj}$$

But by induction, $|\{\text{length } m-1 \text{ paths from } v_i \text{ to } v_k\}| = A_{ik}^{m-1}$, and $\sum_{k=1}^{n} A_{ik}^{m-1} A_k j = A_{ij}^{m}$, so indeed A_{ij}^{m} is the number of paths from v_i to v_j of length m. ■

4.5 Connectedness

Clearly if u is connected to v, and v is connected to w, then u is connected to w. That is, the "connected" relation is *transitive*. It is also *reflexive* since every vertex is by definition connected to itself. If the graph is simple, then obviously u is connected to v iff v is connected to u, so the relation is *symmetric*. That is, "connected" is actually an *equivalence relation* (*cf.* Rosen, section 6.5) on the set of vertices of a simple graph.

A graph is *connected* if there is a path between every pair of distinct vertices. For example, referring back to Figure 4, the empty graph is disconnected, but all others shown are connected.

A subset of the vertices of an undirected graph is a *connected component* of the graph if it consists of precisely all the vertices connected to some single vertex. That is, the connected components of a graph are the *equivalence classes* of the connectedness relation. In particular, every vertex of a simple graph belongs to exactly one connected component, and for any two vertices v_1, v_2, either the component of v_1 is the same as the component of v_2 (when v_1 is connected to v_2), or the two components have no elements in common (when v_1 is not connected to v_2).

Another way to say that a subset of vertices is a connected component is to say that every pair of vertices in the subset is connected, and the subset is *maximal* for this property, that is, no new vertex can be added to the subset which keeps the subset connected.

So a simple graph is connected iff it has exactly one connected component. The empty graph on n vertices has n connected components. A graph with three connected components is shown in Figure 7. We let $\gamma(G)$ be the number of connected components in a simple graph, G.

A False Theorem about Connectivity

If a graph is connected, then every vertex must be connected to some other vertex. Is the converse of this statement true? If every vertex is connected to some other vertex, then is the graph connected? The answer is no. In fact, the graph with three connected components shown in in Figure 7 is a counterexample. So what is wrong with the following proof?

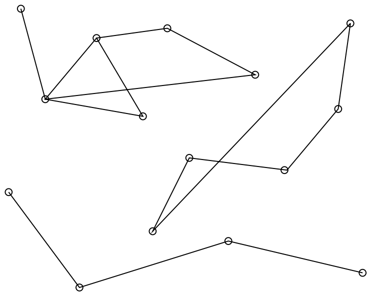


Figure 7: *This is a picture of a graph with 3 connected components.*

False Theorem 4.6 *If every vertex in a graph is adjacent to another vertex, then the graph is connected.*

Nothing helps a false proof like a good picture; see Figure 8.

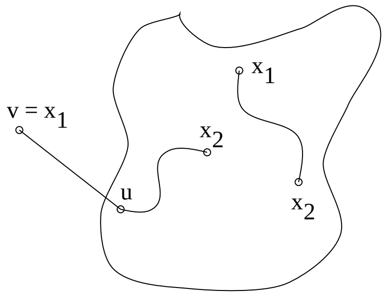


Figure 8: *This picture accompanies the false proof. Two situations are depicted. In one, vertices x_1 and x_2 both are among the vertices of G, and so there is a connecting path by induction. In the second, $v = x_1$ and x_2 is a vertex of G. In this case there is a connecting path because there is an edge from v to u and a path in G from u to x_2 by induction.*

Proof. The proof is by induction. Let $P(n)$ be the predicate that if every vertex in an n-vertex graph is adjacent to another vertex, then the graph is connected. In the base case, $P(1)$ is trivially true because there only one vertex.

In the inductive step, we assume $P(n)$ to prove $P(n+1)$. Start with an $n+1$-vertex graph, G', in which every vertex is adjacent to another vertex. Now take some vertex v away from the graph and let the G be the remaining grpah. By assumption v is connected in G' to one of the n vertices of G; call that one u.

Now we must show that for every pair of distinct vertices x_1 and x_2 in G', there is a path between them. If both x_1 and x_2 are vertices of G, then since G has n vertices, we may assume by induction it is connected. So there is a path between x_1 and x_2. Otherwise, one of the vertices is v (say x_1) and the other, x_2 is in G. But x_2 is connected to u by induction, so there is a path from x_1 to u to x_2 as shown in the figure. ■

The error is in the statement "*since G has n vertices, we may assume by induction it is connected.*" The induction hypothesis does not say that every n-vertex graph is connected, but only, "*if every* *vertex in an n-vertex graph is adjacent to another vertex,* then the graph is connected". For example, if G' is the graph with vertices 1, 2, 3, 4 and edges $\{1,2\}$ and $\{3,4\}$, then removing vertex 1 to form G leaves vertex 2 without an adjacent vertex in G, and we can't conclude by induction that G is connected (which of course it isn't).

The core problem here is called "buildup error" and is a common source of mistakes in inductive proofs. Buildup error occurs when the induction hypothesis $P(n)$ is itself an implication of the form $A(n) \Rightarrow B(n)$. In this instance, $A(n)$ is the statement, "every vertex in an n-vertex graph is adjacent to another vertex" and $B(n)$ is the statement, "the graph is connected".

In the inductive step of a correct proof, we assume $A(n) \Rightarrow B(n)$ in order to prove $A(n+1) \Rightarrow B(n+1)$. In an erroneous proof, we assume $B(n)$ in order to prove $A(n+1) \Rightarrow B(n+1)$. This is precisely what happened; we assumed, "the graph is connected" rather than, "if every vertex in an n-vertex graph is adjacent to another vertex, then the graph is connected".

A True Theorem about Connectivity

If a graph has too few edges, then there cannot be a path between every pair of vertices. More generally, a graph with a very small number of edges ought to have many connected components. (Remember, "many connected components" does not mean "very connected"; rather, it means "broken into many pieces"!) The following theorem generalizes these observations.

Lemma 4.7 *Adding a single edge to a simple graph reduces the number of connected components by at most one. More precisely, let G be a simple graph and G' be G with the addition of a single edge between two vertices of G. Then $\gamma(G') \geq \gamma(G) - 1$.*

Proof. If the new edge is between two vertices which are already connected, then the connectedness relation between vertices remains the same with or without the edge, so G and G' have exactly the same connected components and $\gamma(G') = \gamma(G)$. If the new edge is between vertices v_1 and v_2 which are not connected in G, then the component of v_1 and the component of v_2 are distinct in G and become one component of G'. The components of G which contain neither v_1 nor v_2 remain as connected components of G'. The effect is that the number of components in G' is one less than in G, *viz.*, $\gamma(G') = \gamma(G) - 1$. ∎

Corollary 4.8 *For any simple graph $G = (V, E)$,*

$$\gamma(G) \geq |V| - |E|.$$

For example, a graph with 0 edges and n vertices has at least $n - 0 = n$ connected components. A graph with 100 vertices and 35 edges must have 100 - 35 = 65 connected components.

Proof. By induction on $|E|$. (This may seem odd, because for $|E| > |V|$, the theorem says that the number of connected components is greater than some negative number! This is useless, but certainly true, so there is no harm.)

The induction hypothesis will be $P(k)$: any graph $G = (V, E)$ with k edges has at least $|V| - k$ connected components.

In the base case, $P(0)$ holds because in a graph with 0 edges, each vertex is a connected component, so $\gamma(G) = |V| \geq |V| - 0$.

In the inductive step, let $G' = (V', E')$ be a graph with $k+1$ edges. We want to prove that $\gamma(G') \geq |V'| - (k+1)$.

Pick some edge of G' and let G be G' with the chosen edge removed. By the lemma above,

$$\gamma(G') \geq \gamma(G) - 1.$$

By induction we may assume that $P(k)$ is true, and since G has k edges, we have

$$\gamma(G) \geq |V| - k.$$

But $|V| = |V'|$, so

$$\gamma(G') \geq \gamma(G) - 1 \geq (|V'| - k) - 1 = |V'| - (k+1).$$

■

Since a graph G is connected iff $\gamma(G) = 1$, we conclude immediately:

Corollary 4.9 *If a simple graph $G = (V, E)$ is connected, then*

$$|E| \geq |V| - 1.$$

5 Trees

A simple graph is *acyclic* iff it has no simple cycles. Rosen §8.1 defines a *tree* to be a simple, connected, acyclic graph. In lecture, we defined a tree to be a simple connected graph (V, E) in which $|E| = |V| - 1$. The following theorem shows that these two characterizations of trees, and two additional ones, are all equivalent.

We did not cover this theorem in lecture or recitation, but include it here for the record. It will not appear on exams in 6.042 this term.

Theorem 5.1 *For any simple graph $G = (V, E)$, the following are equivalent:*

1. *G is connected and $|E| = |V| - 1$.*
2. *G is connected, but removing any edge from G leaves a disconnected graph.*
3. *G is connected and acyclic.*
4. *There is a unique simple path between any two distinct vertices of G.*

Of course it would be a waste of effort to prove that each statement implied all the others – twelve different implications. Instead, we prove that each implies the next, and the fourth implies the first – only four implications. Some of the implications among these properties are harder to prove directly than others, so the order in which they are listed will affect the simplicity of the four direct implications to be proved. You might wonder how to pick an order which leads to simple proofs. There was no trick used here, just trial and error (over several hours) to find an order which allowed the simplest proofs – and there may well be a still simpler one we overlooked.

Proof.

- (1 implies 2) This follows immediately from Corollary 4.9.
- (2 implies 3) By contradiction.

  Suppose proposition 2 holds, but G has a simple cycle. Then removing the first edge in the cycle leaves a simple path connecting all the vertices in the cycle. This implies that the connected component of the vertices in the cycle is the same after the edge is removed as it was in G. Since there was only one connected component to begin with, there is still only one after the edge is removed. That is, the graph is still connected, contradicting proposition 2.
- (3 implies 4) By contradiction.

  Suppose proposition 3 holds, but there are distinct simple paths between two vertices of G. That is, there are vertices $u \neq v$ and distinct simple paths P_1 with vertices $u, w_1, \ldots, w_n, v$ and P_2 with vertices $u, w'_1, \ldots, w'_{n'}, v$. Among all such u, v, P_1, P_2, we can, by the Well-ordering Principle, choose some for which P_1 is shortest.

  Suppose P_1 a single edge, that is $P_1 = \{u, v\}$. If P_2 started with this same edge, then P_2 without its first edge would be a simple cycle from v to v, contradicting proposition 3. On the other hand, if P_2 did not start with $\{u, v\}$, then since $\{u, v\} = \{v, u\}$, the path starting with this edge followed by the edges in P_2 form a simple cycle from v to v, again contradicting proposition 3.

  Hence, P_1 is of length greater than one, and $1 \leq n \leq n'$. Moreover, the first edge of P_1 must differ from the first edge of P_2, since otherwise $w_1 = w'_1$ and P_1 and P_2 without their first edges would be distinct paths between w_1 and v, contradicting the minimality of the length of P_1.

  Now we claim that no vertex w_i internal to P_1 is the same as any vertex w'_j internal to P_2. This follows because if $w_i = w'_j$, then the two paths $u, w_1, \ldots, w_i$ and $u, w'_1, \ldots, w'_j$ between u and w_i would be distinct because their first edges differ, and the first path would be shorter than P_1, again contradicting the minimality of P_1. So no w_i can equal any w'_j. Hence the concatenation of all but the last edge of P_1 with the reversal of P_2, namely,

  $$u, w_1, \ldots, w_n, w'_{n'}, \ldots, w'_1, u$$

  is a simple cycle, contradicting proposition 3.
- (4 implies 1) By strong induction on $|V|$.

  (Base case: $|V| = 1$) Proposition 1 is immediate if G has one vertex.

(Induction) Suppose proposition4 implies proposition 1 for all graphs with $1 \leq k \leq n$ vertices. Let G be a graph with $n+1$ vertices satisfying proposition4. We must show that proposition 1 holds for G.

Since G has $n+1 \geq 2$ vertices, and any two vertices are connected by a simple path, G certainly has at least one edge. Let G' be the graph which is left after removing some edge, $\{u, v\}$, from G.

If there was still a path between u and v in G', then by Lemma 4.4 there would a simple path between u and v in G'. But then in G, this path and the edge $\{u, v\}$ would be distinct simple paths between u and v, contradicting proposition 4. So G' cannot be connected. Let $G_i = (V_i, E_i)$ for $1 \leq i \leq n \geq 2$ be the set of connected components of G'.

Now each connected component G_i still has unique simple paths and has fewer vertices than G, so by strong induction $|E_i| = |V_i| - 1$. Now we have

$$\begin{array}{rclclcl} |V| & = & \sum_{i=1}^{n} |V_i| & = & \sum_{i=1}^{n} (|E_i| + 1) & & \\ & = & (\sum_{i=1}^{n} |E_i|) + n & = & |E'| + n & & \\ & \geq & |E'| + 2 & = & (|E| - 1) + 2 & = & |E| + 1. \end{array}$$

That is, $|E| \leq |V| - 1$. But since G is connected, we have from Corollary 4.9 that $|E| \geq |V| - 1$. So $|E| = |V| - 1$.

■