Graphs

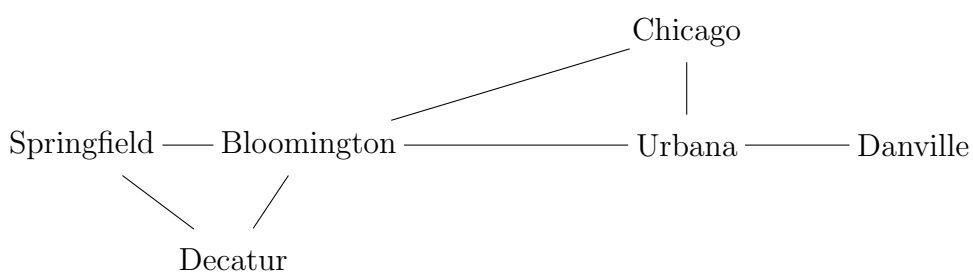
Margaret M. Fleck

14 April 2010

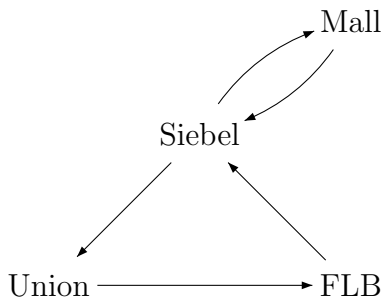
This lecture introduces graphs (section 9.1. and 9.2 of Rosen).

1 Graphs

Graphs are a very general class of object, used to represent a wide variety of relationships and complex objects in computer science. A simple example of a graph is the map of roads connecting cities shown below. In this example, each edge can be traversed in both directions. This is called an undirected graph.



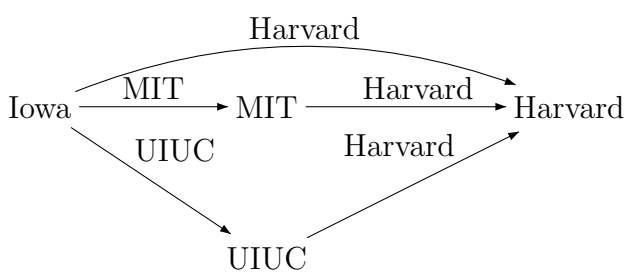
Sometimes transportation along certain routes is only one-way. For example, the examples below shows a possible (but non-existent) circular bus route serving three buildings on our campus. This is a directed graph. When a link works in both directions, you have to show this explicitly, as in the case of the busses to/from the Mall.



An undirected graph has "2-way" edges, which can be traversed in both directions. For example, most streets or facebook friend relationships (which must be accepted by the other person). A directed graph has one-directional edges. For example, twitter following is a directed relationship. A celebrity may be followed by many people, few of whom the celebrity follows.

Graphs can also represent more abstract relationships among objects. For example, web pages contain hyperlinks to other web pages. Each of these links has some anchor text, which labels the page at the other end of the link. Web search systems such as Google analyze the structure of these labelled links, as well as the text on each page, to determine which pages are the best ones to return for a given search keyword.

These hyperlink relationships are typically not symmetrical: a link in one direction doesn't imply a link in the opposite direction. For example, my web page might point to Obama's web page, but the reverse link is unlikely. In fact, analysis of web pages suggests there are certain classes of pages that have mostly incoming links or mostly outgoing links. A running joke at most places other than Harvard is that other people compare to Harvard but Harvard ignores everyone else. So we might have a hyperlink graph like



In this example, notice that I've put labels on the graph edges, showing the anchor text for each link. In this class, we won't make much use of labelled edges, but you'll see them a lot in future classes e.g. automata theory.

2 Some definitions

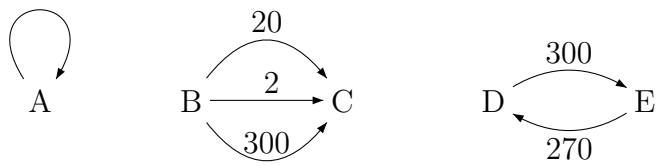
Formally, a (finite) graph consists of a finite set of vertices V and a finite set of edges E. There are infinite graphs, but we won't see any in this class. For a directed graph, E is a set of ordered pairs of edges, i.e. a subset of $V \times V$.

There's several ways to formalize the edges for an undirected graph. The approach chosen by Rosen is to make each edge a set $\{v_1, v_2\}$ containing two vertices. In this case, E is a subset of $\{x, y \mid x, y \in V\}$, which is a subset of $\mathbb{P}(E)$. Or, if we don't allow self-loops,, E is a subset of $\{x, y \mid x, y \in V \text{ and } x \neq y\}$. Alternatively, we could use ordered pairs but have a convention that (v_1, v_2) is in E if and only if (v_2, v_1) is in E. We won't need to worry about this level of detail this term.

There are many terms for different types of graphs, not entirely standard across authors. Rather than memorizing a large set of terms, it's best to concentrate on our design options:

- Is the graph directed or undirected?
- Can it contain self-loops, i.e. edges that go from a node back to the same node?
- Can it contain multi-edges, i.e. two different edges with the same starting and ending nodes?

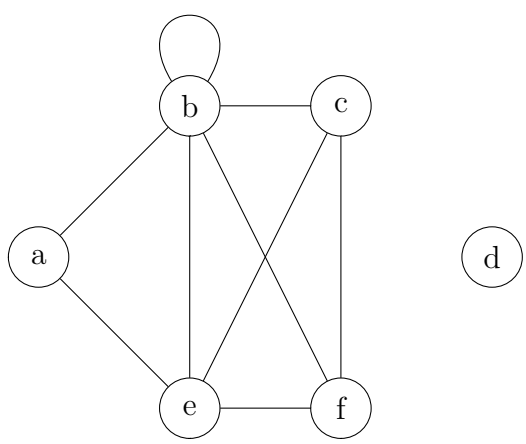
A self loop and a multi-edge are illustrated below. The labels on the multi-edge might be, for example, the costs for various ways to get between two cities B and C (e.g. car, airplane, hitchhiking). The righthand graph might be how long it takes a plane to fly between cities D and E. Notice that the edges connecting D and E are not multi-edges, because they go in opposite directions.



Which of these options are allowed depends on the needs of the application. Self-loops are often allowed in directed graphs, and often forbidden in undirected graphs. Multi-edges are usually allowed only in applications where labels on the edges make the different edges differ from one another in interesting ways.

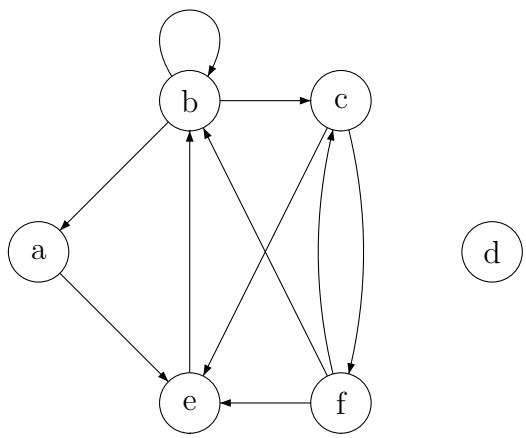
3 Degrees

In an undirected graph, the degree of a vertex v, written $deg(v)$ is the number of edges incident to v (i.e. having v as an endpoint). Self-loops, if you are allowing them, count twice. For example, in the following graph, a has degree 2, b has degree 6, d has degree 0, and so forth.



In a directed graph, each vertex has an in-degree (written $deg^-(v)$) which is the number of incoming edges, and an out-degree (written $deg^+(v)$) which is the number of outgoing edges. The degree $deg(v)$ is then the sum of the in-degree and the outdegree of v. For example, in the following graph, f has

in-degree 1 and out-degree 3. The previous graph is called the "underlying undirected graph" for this one, i.e. it's what you get if you remove the directionality from each edge (and merge pairs of edges joining the same two nodes in different directions).



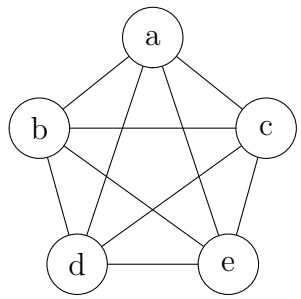
In an undirected graph, each edge contributes to two vertex degrees. So the sum of the degrees of all the vertices is twice the number of edges. This is called the Handshaking Theorem and can be written as

$$\sum_{v \in V} deg(v) = 2|E|$$

4 Complete graphs

A simple graph is undirected, has no self-loops and no multi-edges. We'll see five special types of graphs, all of which are simple graphs.

The complete graph on n vertices ($n \geq 1$) shorthand name K_n, is a graph with n vertices in which every vertex is connected to every other vertex. K_5 is shown below. In principle, you could have a graph with no vertices, so there could be a K_0, but we won't worry about that this term.



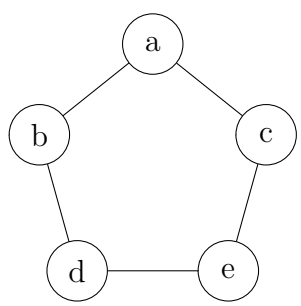
To calculate the number of edges in K_n, think about the situation from the perspective of the first vertex. It is connected to $n-1$ other vertices. If we look at the second vertex, it has $n-2$ more connections. And so forth. So we have $\sum_{k=1}^{n} n-k = \frac{n(n-1)}{2}$ edges. Another way to do this calculation is to notice that there are $\binom{n}{2}$ ways to pick a subset of two vertices. $\binom{n}{2}$ is also equal to $\frac{n(n-1)}{2}$.

5 Cycles and wheels

Suppose that we have n vertices named $v_1, \ldots, v_n$. Then the cycle C_n ($n \geq 3$) is the graph with these vertices and edges

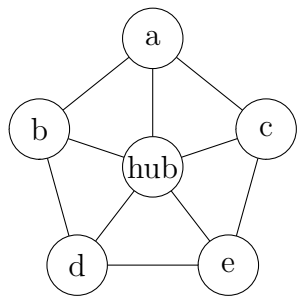
$$E = \{\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{n-1}, v_n\}, \{v_n, v_1\}\}$$

So C_5 looks like



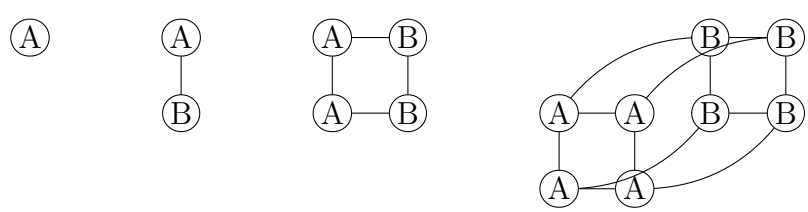
C_n has n vertices and also n edges. Cycle graphs often occur in networking applications. They could also be used to model games like "telephone" where people sit in a circle and communicate only with their neighbors.

The wheel W_n is just like the cycle C_n except that it has an additional central "hub" node which is connected to all the others. Notice that W_n has $n+1$ nodes (not n nodes). It has $2n$ edges. For example, W_5 looks like



6 Hypercubes

An n-cube or a hypercube Q_n is the graph of the corners and edges of an n-dimensional cube. It is defined recursively as follows (for any $n \in \mathbb{N}$):

1. Q_0 is a single vertex with no edges
2. Q_n consists of two copies of Q_{n-1} with edges joining corresponding vertices.

That is, each node v_i in one copy of Q_{n-1} is joined by an edge to its clone copy v_i' in the second copy of Q_{n-1}. Q_0, Q_1, Q_2, and Q_3 look as follows. The node labels distinguish the two copies of $Qn-1$