Graph isomorphism/connectivity

Margaret M. Fleck

16 April 2010

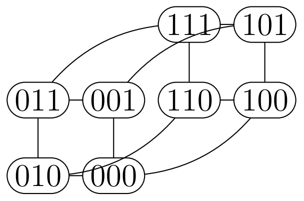
This lecture finishes our coverage of basic graph concepts: isomorphism, paths, and connectivity. It shows the basic ideas without covering every possible permutation of these ideas e.g. for different types of graphs.

1 Hypercubes

Recall from last class that the 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.

The hypercube defines a binary coordinate system. To build it, we label nodes with binary numbers, where each binary digit corresponds to the value of one coordinate.



Q^n has 2^n nodes. To compute the number of edges, we set up a recurrence:

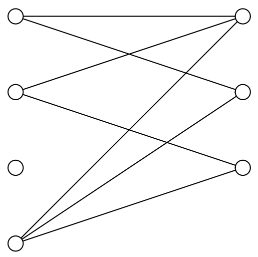
1. $E(0) = 0$
2. $E(n) = 2E(n-1) + 2^{n-1}$

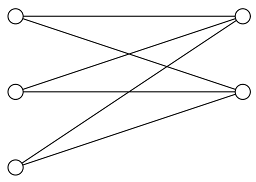
The 2^{n-1} term is the number of nodes in each copy of Q^{n-1}, i.e. the number of edges required to join corresponding nodes.

2 Bipartite graphs

The last special type of graph is a bipartite graph. A graph $G = (V, E)$ is bipartite if we can split V into two non-overlapping subsets V_1 and V_2 such that every edge in G connects an element of V_1 with an element of V_2. That is, no edge connects two nodes from the same part of the division. Bipartite graphs often appear in matching problems, where the two subsets represent different types of objects, e.g. matching a group of women with a group of male study partners.

The complete bipartite graph $K_{m,n}$ is a bipartite graph with m nodes in V_1, n nodes in V_2, and which contains all possible edges that are consistent with the definition of bipartite. The diagram below shows a partial bipartite graph on a set of 7 nodes, as well as the complete bipartite graph $K3, 2$.





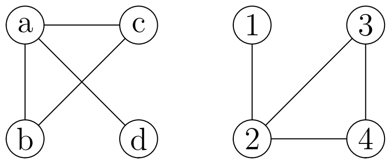
The complete bipartite graph $K_{m,n}$ has $m + n$ nodes and mn edges.

3 Isomorphism

Two graphs are called isomorphic if they have the same structure, even though their vertices and edges might have been positioned differently in 3D or in a 2D picture. Let's look at how this idea is made formal.

Recall that a simple graph is an undirected graph with no self-loops (no edges from a vertex bad to itself) and no multi-edges (multiple edges between the same pair of vertices). Suppose that $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are simple graphs. Then G_1 and G_2 are isomorphic if there is a bijection $f : V_1 \to V_2$ such that vertices a and b are joined by an edge if and only if $f(a)$ and $f(b)$ are joined by an edge.

For example, consider the following two graphs:



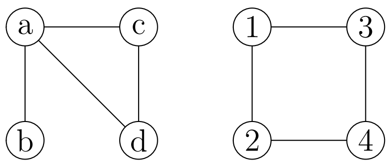
These two graphs are isomorphic. We can prove this by defining the function f so that it maps 1 to d, 2 to a, 3 to c, and 4 to b. The reader can then verify that edges exist in the left graph if and only if the corresponding edges exist in the right graph.

To prove that two graphs are not isomorphic, we could walk through all possible functions mapping the vertices of one to the vertices of the other. However, that's a huge number of functions for graphs of any interesting size. An exponential number, in fact. Instead, a better technique for many examples is to notice that a number of graph properties are "invariant," i.e. preserved by isomorphism.

- The two graphs must have the same number of vertices and the same number of edges.
- For any vertex degree k, the two graphs must have the same number of vertices of degree k.

- Any subgraph of the first graph must have a matching subgraph somewhere in the second graph. (We would normally choose a small subgraph.)

We can prove that two graphs are not isomorphic by exhibiting one example of a property that is supposed to be invariant but, in fact, differs between the two graphs. For example, in the following picture, the lefthand graph has a vertex of degree 3, but the righthand graph has no vertices of degree 3, so they can't be isomorphic.

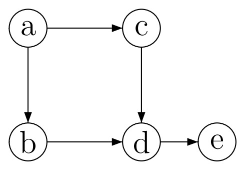


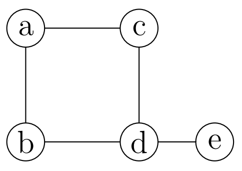
4 Paths

In an undirected graph, a path of length k from vertex a to vertex b is a sequence of edges that connect end-to-end $(v_1, v_2), (v_2, v_3), (v_3, v_4), \ldots, (v_{k-1}, v_k)$, where $v_1 = a$, $v_2 = b$. A single vertex a might or might not be considered a path of length from a to a. This depends on the author.

This definition works for a directed graph as well. Just beware that your path must follow the arrows on the edges.

For example, in the lefthand graph, there is a path from a to d: $(a, c), (c, d), (d, e)$. But there isn't a path from e back to a, because the arrows go the wrong way. In the righthand undirected graph, both paths exist.



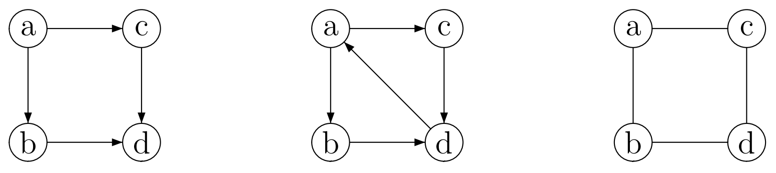


5 Connectivity

An undirected graph G is connected if there is a path between every pair of nodes in G. That is, for any nodes a and b in G, there is a path from a to b. If G is directed, we have two notions of connectivity:

- G is strongly connected if there is a (directed) path between any pair of vertices in G.
- G is weakly connected if its underlying undirected path (i.e. remove the directions from all the edges) is connected.

If a and b are two nodes in G, strong connectivity requires that there be a path from a to b and also a path from b to a. So the lefthand graph in the following picture is not connected, because there's no way to get from d to a. This graph is weakly connected, because its underlying undirected graph (on the right in the figure) is connected. The directed graph in the middle, with an extra edge from d to a, is strongly connected.



If we have an undirected graph G, and not all the vertices of G are connected to one another, we can divide G into "connected components."

Each connected component contains a maximal set of vertices that are all connected to one another, plus all their edges. ("Maximal" means to choose the biggest groups of vertices for which this is true.) So, the following graph has two connected components, one containing vertices a,b,c, and d and the second containing vertices e and f.