CMPSCI 250: Introduction to Computation

Lecture #15: Graphs and Paths
David Mix Barrington
26 March 2013

Graphs

- Modeling Connections
- Different Types of Graphs
- Some Terminology
- Bipartite Graphs
- Hall's Theorem
- Operations on Graphs
- Representations and Isomorphism

Modeling Connections

- Many situations in the real world may be modeled as a set of **things**, some of which are **connected** to others. Cities are connected by roads, airports by flights, Facebook customers by links to their friends, pages on the web by links to other pages. **Graph theory** is the branch of mathematics that deals with such situations.

- Often we are interested only in the fact of the connection between two things, the binary relation that says “x and y are connected”. In this case a graph is just a way of drawing a binary relation, with a dot or **vertex** for each element of the base set and a line or **edge** from x to y whenever x and y are connected. In some cases we are modeling a binary relation that is not symmetric, so that x could be connected to y without y being connected to x. In this case we draw an arrow or **arc** or **directed edge** from x to y.

- In other cases we add **labels** to the arcs or edges, giving distances, costs, or other information about that particular connection.

Different Types of Graphs

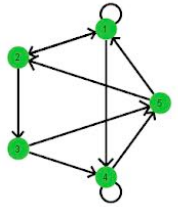
- The different types of connections give us several different types of graphs -- Rosen defines six. When we model a situation with a graph, we must decide whether connections must be symmetric and thus whether our edges are directed. We also need to decide whether it makes sense to have more than one **parallel edge** with the same endpoints, or to have edges from a vertex to itself, called **loops**. Rosen's definitions are in a table on page 644.

- In Rosen's terminology, a **simple graph** has undirected edges but no parallel edges or loops. Two other types have multiple edges: a **multigraph** may have parallel edges but not loops, and a **pseudograph** may have either parallel edges or loops.

- A **simple directed graph** has directed edges but no parallel edges or loops. A **directed multigraph** may have either parallel edges or loops, and a **mixed graph** may have edges that are either directed or undirected.

iClicker Question #1: Graph Examples

- **What type of graph** is pictured at right?
- (a) a simple graph
- (b) a multigraph
- (c) a simple directed graph
- (d) a directed multigraph



Some Terminology

- If x is a vertex in an undirected graph, another node y is x's **neighbor** if there is an edge between x and y. The **neighborhood** of x is the set of all its neighbors, and the neighborhood of a set of vertices A is the set of all nodes that are neighbors of any vertex in A.

- The **degree** of a vertex in an undirected graph is its number of neighbors. In a directed graph, a vertex has both an **in-degree** (number of arcs into it) and an **out-degree** (number of arcs out of it). Rosen refers to vertices of degree 0 as **isolated** and vertices of degree 1 as **pendant**.

- The **handshaking theorem** says that the sum of the degrees of all the vertices is equal to twice the number of edges. This follows from the fact that each edge contributes one to the degree of each of its endpoints, and thus contributes two to the sum. This result works for multiple edges as well, and works for loops as long as we count the loop twice for the degree.

iClicker Question #2: The Handshake Theorem

- We say that a vertex of a simple graph is odd if it has an odd number of neighbors, and even if it has an even number of neighbors. Which of these four situations is **impossible** in a simple graph?

- (a) There is an even number of even vertices.

- (b) There is an odd number of even vertices.

- (c) There is an even number of odd vertices.

- (d) There is an odd number of odd vertices.

Families of Named Graphs

- To have examples easily at hand, it's useful to give names to some standard families of graphs.

- If n is a positive integer, the **complete graph K_n** has n vertices and an edge between every pair of distinct vertices. K_1 has no edges, K_2 has 1, K_3 has 3, and in general K_n has $n(n-1)/2$.

- If $n \geq 3$, the **cycle C_n** has n vertices $v_1,\ldots, v_n$ and n edges, with each v_i being connected to v_{i+1} and v_n being connected to v_1, so that every vertex has degree 2. The wheel W_n has n+1 vertices, n of them making a copy of C_n and the last connected to each of the first n. It has 2n edges: the first n vertices each have degree 3 and the last has degree n.

- The **n-dimensional hypercube Q_n** has 2^n vertices, one for every binary string of length n, and a edge between the vertices for strings u and v if and only if u and v differ in exactly one bit.

Bipartite Graphs

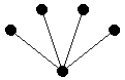
- Suppose that A and B are two sets and we have a relation $R \subseteq A \times B$, so that R(a, b) is true or false for each choice of a in A and b in B. We can draw a graph where there are vertices for each element of A and each element of B, and an edge between a and b whenever R(a, b) is true.

- This graph has the property that every edge has one endpoint in A and the other in B. If we look at any undirected graph, we say that it is **bipartite** if it is possible to divide the vertex set V into two subsets A and B, such that all edges have this property.

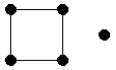
- It's easy to test whether a simple graph is bipartite. We make a **greedy two-coloring** of the vertices, beginning with an arbitrary vertex that we color red. We then color the neighbors of red vertices blue, and neighbors of blue vertices red, as long as we can. If we fail, then the graph is not bipartite. If we finish and there are uncolored vertices left, we start again with one of them. A theorem says a graph is bipartite if and only if it has no **odd cycle**.

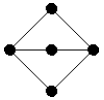
iClicker Question #3: Which is Not Bipartite?

- Here are four graphs, three of which are bipartite. **Which graph is not bipartite?** That is, which graph's vertices **cannot be two-colored** so that no edge connects two vertices of the same color?

A 

 B

C 

 D

Hall's Theorem

- A matching in a bipartite graph is a set of edges that represents a one-to-one function from A to B -- it must have exactly one edge for each element of A and no two edges may share an endpoint in B. **Hall's Theorem**, proved by Philip Hall around 1928, is a criterion for when a matching exists.

- If X is any nonempty subset of A, let N(X) be its neighborhood. We say that X is a **bottleneck** if |N(X)| < |X|, and say that X is **tight** if |N(X)| = |X|. Hall's Theorem says that a matching exists if and only if there is no bottleneck.

- It's easy to see that no matching can exist if there *is* a bottleneck. The difficulty is to prove that a matching *must* exist if there isn't one. We do this by a strong induction. Our statement P(k) is "if a subset X of A, with k elements, has no bottleneck subset, then it has a matching".

- For the base case, if |X| = 1 and X is not a bottleneck, it has a matching.

More of the Proof of Hall's Theorem

- So now we assume P(j) for all j with $j \leq k$. That is, if any nonempty set of size at most k has no bottleneck subset, it has a matching. We then need to prove P(k+1) to complete the strong induction. This says that if $|X| = k+1$, and X has no bottleneck subset, then X has a matching. We prove this with two cases.

- First assume that X has no proper tight subset. Pick any element z of X, let b be one of its neighbors in B, and remove both z and b from consideration. The resulting set $X \setminus \{z\}$ has k elements, so by the IH it has a matching unless it has a bottleneck. But it can't have a bottleneck, because we removed only one element of B and every subset had an extra neighbor in B because it was not tight. So we have this matching on $X \setminus \{z\}$, which becomes a matching on X when we reinsert z and connect it to b. We've proved P(k+1) for this case.

Finishing the Proof of Hall's Theorem

- The other case is when X has a proper tight subset Y.
- Since Y is a proper subset of X, it has at most k elements. It has no bottleneck because if it did, X would have a bottleneck. So by the IH, Y has a matching.
- Remove Y and its matched elements of B from consideration. We have left a proper subset Z of X, and some remaining elements of B. We need to prove that Z has no bottleneck -- then it has a matching which we can combine with Y's matching to get a matching on X.
- Suppose that W were a bottleneck in Z. Consider the set $W \cup Y$ with the partners of Y's elements returned to consideration. If W had fewer than $|W|$ neighbors without Y's partners, then $W \cup Y$ would have had fewer than $|W \cup Y|$ neighbors originally, and been a bottleneck in X. But X had no bottlenecks.

Operations on Graphs

- There are various operations and relations defined on graphs, of which we will define a few. The word **subgraph** has a variety of definitions -- Rosen says that if G = (V, E) is a graph, then H = (V', E') is a subgraph of G if V' $\subseteq$ V and E' $\subseteq$ E. Note that the edges in E' may only involve the vertices in V', for H to be a properly defined graph.

- If G = (V, E) is a graph and V' $\subseteq$ V, the **subgraph induced by** V' has V' as its vertices and has *all* the edges of E that use only vertices in V'.

- If G and H are graphs with the same vertex set V, we can make the union or intersection of the two graphs by unioning or intersecting their edge sets.

- More generally, we can take the union or intersection of any two graphs by taking the union or intersection of both the vertex sets and edge sets.

Representations of Graphs

- Representing a graph in a computer is essentially a matter of representing the edge predicate. Different data structures to do this have different advantages.

- If G = (V, E) and |V| = n, the **adjacency matrix** of G is an n by n matrix whose rows and columns are indexed by elements of v. The (u, v) entry of the matrix is 1 if there is an edge from u to v and 0 otherwise. More generally, in a multigraph the (u, v) entry of the matrix gives the *number* of edges from u to v.

- In CMPSCI 311, you'll more often represent graphs by an **adjacency list**, where for each vertex u you have a list of the vertices v such that there is an edge from u to v. If the graph is sparse (has many fewer than n^2 edges), an adjacency list is much smaller than an adjacency matrix and allows for many faster algorithms.

- There's also the incidence matrix, with rows for vertices and columns for edges. The (v, e) entry is 1 if v is an endpoint of e, and 0 otherwise.

The Idea of Isomorphism

- It's often possible for two objects to be **equal** in all essential respects without being **identical** as data structures. In Java classes we write an `equals` method to tell when this happens. In mathematics we often define what it means for two objects of the same type to be **isomorphic**.

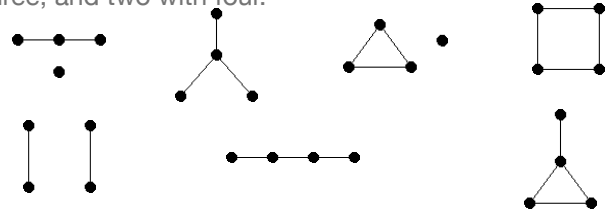
- In general an isomorphism is a one-to-one, onto function from one object to another that preserves the important properties of the object. For example, an isomorphism of sets is a bijection, and two sets are isomorphic if they have the same cardinality.

- In linear algebra, an isomorphism is a linear map (homomorphism) from one vector space to another that is also a bijection. Two vector spaces over the same field are isomorphic if they have the same dimension.

Graph Isomorphism

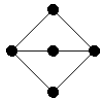
- Let G = (V, E) and H = (V', E') be two graphs. A function f is an **isomorphism** of G and H if it is a bijection from V to V', and for any vertices u and v in V, there is an edge in Efrom u to v if and only if there is an edge from f(u) to f(v) in E'. We can think of an isomorphism as a renaming of the vertices of G that makes the resulting graph identical to H.

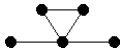
- If G and H are isomorphic graphs, they must have the same number of vertices and same number of edges. For graphs of 1, 2, or 3 vertices, any pair of graphs with the same number of both nodes and edges are isomorphic. But for 4-node simple graphs we have two different graphs with two edges, three with three, and two with four.



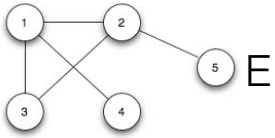
iClicker Question #4: Graph Isomorphism

- Here are five simple graphs. Exactly one of the graphs A, B, C, and D is isomorphic to graph E. **Which one** is isomorphic to E?

A 

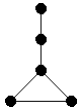


B



E

C 



D