CS3203 #10

6/28/04

Janak J Parekh

Administrivia

- I'm going to make HW#5 smaller, and due this Friday
  – Simpler administratively than to push it off after July 4^{th}
- Final exam on Wednesday
  – Covers everything through this lecture

Bipartite graphs

- A simple graph G is called *bipartite* if its vertex set V can be partitioned into two disjoint sets, V_1 and V_2, such that every edge in the graph connects a vertex in V_1 and a vertex in V_2.
  – Is C_6 bipartite?
  – Is K_3 bipartite?
  – $K_{m,n}$ is a complete bipartite graph partitioned into *m* and *n* vertices.

Special applications of special types of networks

- Local area networks – star, ring, hybrid (starred ring), bus topologies
- Interconnection networks for parallel computation
  - K_n – but expensive and limited
  - Linear array – need lots of intermediate hops?
  - Mesh network – grid (2d array); communication requires $O(\sqrt{n})$ intermediate links
  - Hypercube

Generating new graphs from old graphs

- A subgraph of a graph $G=(V,E)$ is a graph $H=(W,F)$ where $W \subseteq V$ and $F \subseteq E$.
- The union of two simple graphs $G_1=(V_1,E_1)$ and $G_2=(V_2,E_2)$ is the simple graph with vertex set $V_1 \cup V_2$ and edge set $E_1 \cup E_2$, and the graph is denoted by $G_1 \cup G_2$.

Representing graphs and graph isomorphisms

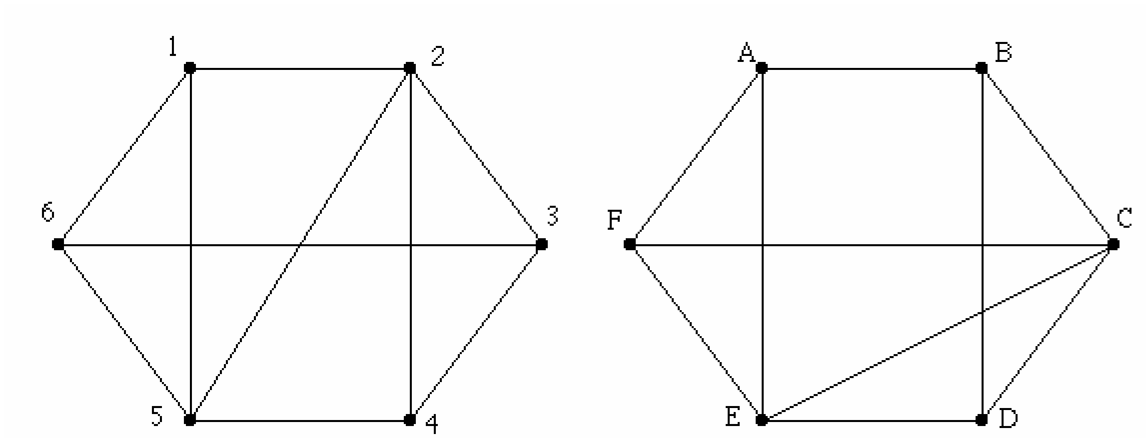
- Adjacency list
  - Simple table, page 557
  - For undirected graph, for each vertex list adjacent vertices
  - For directed graph, list initial vertex and terminal vertices associated with it
  - Sometimes cumbersome, so…
- Adjacency matrix
  - It's an n x n zero-one matrix with 1 as its (i,j)th entry if v_i and v_j are adjacent, or 0 otherwise.
  - For directed graphs, if (v_i,v_j) is an edge in G
  - Symmetric for simple graphs
  - Typically sparse if there aren't many edges, which may be inefficient
  - For more complex graphs, can use values > 1
- Incidence matrix
  - n x m matrix, n vertices, m edges
  - $M = [m_{ij}]$, where $m_{ij} = 1$ when e_j is incident with $v_{i,}$ 0 otherwise

Isomorphism

- Simple graphs G_1 and G_2 are isomorphic if there is a 1-to-1 and onto function *f* from V_1 to V_2 with the property that *a* and *b* are in G_1 if and only if f(a) and f(b) are adjacent in G_2, for all *a* and *b* in V_1. Such a function is called an *isomorphism*.
- Often more difficult than you'd initially imagine.
- Sometimes can use clues (# of vertices, # of edges, vertex degrees) to help decide if something is not isomorphic, but if they're the same, then you have to figure out another way
- Examples on page 561
- How about…

Example

- Is the following graph isomorphic?



  - Yes: (A,B,C,D,E,F) = (6,3,2,4,5,1)
- Strategies
  - Subgraphs
  - Invariants
    - Degree sequences
    - Correspondences
  - Can sometimes use adjacency matrix to demonstrate isomorphism
  - By hand!

Connectivity

- Now that we've defined graphs, we're concerned with their traversal
- We define a path of length n from u to v in G as a sequence of *n* edges $e_1, \ldots, e_n$ such that $f(e_1) = \{x_0,x_1\}$, $f(e_2) = \{x_1,x_2\}, \ldots, f(e_n) = \{x_{n-1}, x_n\}$, where $x_0 = u$ and $x_n = v$.
  - When the graph is simple, we can denote this by a sequence of vertices $x_0, \ldots, x_n$.
  - A path is a *circuit* if it starts and ends at the same vertex *and* has length greater than zero
  - The path/circuit "passes through" the vertices/"traverses the edges".
  - A path or circuit is simple if it does not contain the same edge more than once. (Differences about duplicating vertices…)
- For directed (multigraphs), a path of length n from u to v in G as a sequence of *n* edges $e_1, \ldots, e_n$ such that $f(e_1) = (x_0,x_1), f(e_2) = (x_1,x_2), \ldots, f(e_n) = (x_{n-1}, x_n)$, where $x_0 = u$ and $x_n = v$.
  - When no multiple edges, we can denote this by a sequence of vertices $x_0, \ldots, x_n$.
  - A path of length greater than zero that begins and ends at the same vertex is a *circuit* or *cycle*.
  - A path or circuit is simple if it does not contain the same edge more than once. (Unclear about duplicating vertices!)

Examples…

- “Degrees of separation”
  - Design a graph with people, and an edge linking them if they know each other
  - Proposed that most pairs of people are linked by a small chain of people, perhaps five or fewer (“six degrees of separation”), which would imply a very short, bushy graph
  - Erdos number of a mathematician *m* is the length of the shortest path between m and the vertex representing Erdos, with edges representing “written papers with”
  - Bacon number of an actor c is the length of the shortest path between c and Bacon, where an edge represents “having acted with”

Connectedness in undirected graph

- An undirected graph is called *connected* if there is a path between every pair of distinct vertices of the graph.
  - There is a simple path between every pair of distinct vertices of a connected undirected graph
  - A graph doesn't need all the vertices to be connected!
- A graph that is not connected is the union of two or more connected subgraphs ("connected components"), each pair of which has no vertex in common.
- If removal of a vertex and all edges incident with it produces a subgraph with more connected components than the original graph, it's a cut vertex/articulation point. Similarly, cut edges or bridges represent edges whose removal disconnects the graph.

Connectedness in directed graphs

- A directed graph is *strongly connected* if there is a path from a to b and from b to a whenever *a* and *b* are vertices in the graph.
- A directed graph is *weakly connected* if there is a path between every two vertices in the underlying undirected graph.
  - Graph is “one piece”

Isomorphism, counting

- The extistence of a simple circuit of a particular length is a useful invariant to demonstrate nonisomorphism
  - Figure 6 on page 573
  - Paths can be used to construct mappings that may be isomorphisms
- Let G be a graph with adjacency matrix **A** with respect to the ordering $v_1, v_2, \ldots, v_n$. The number of different paths of length *r* from v_i to v_j, where *r* is a positive integer, equals the (i,j)th entry of $\mathbf{A}^{r}$.
  - We're not going to do this, too annoying by hand

Eulerian circuits and paths

- Motivated by the Konigsberg bridge problem
  - Was divided into four sections by the branches of the Pregel river
  - Seven bridges connected these regions in the 18th century (page 578)
  - Is it possible to start at point, wander across all the brdges exactly once, and return to the starting point?
- Euler found the answer, and generalized for graphs in general
  - An Euler circuit in graph G is a simple circuit containing every edge of G.
  - An Euler path in G is a simple path containing every edge of G.
- Examples, page 578

Conditions for Eulerian circuits and paths

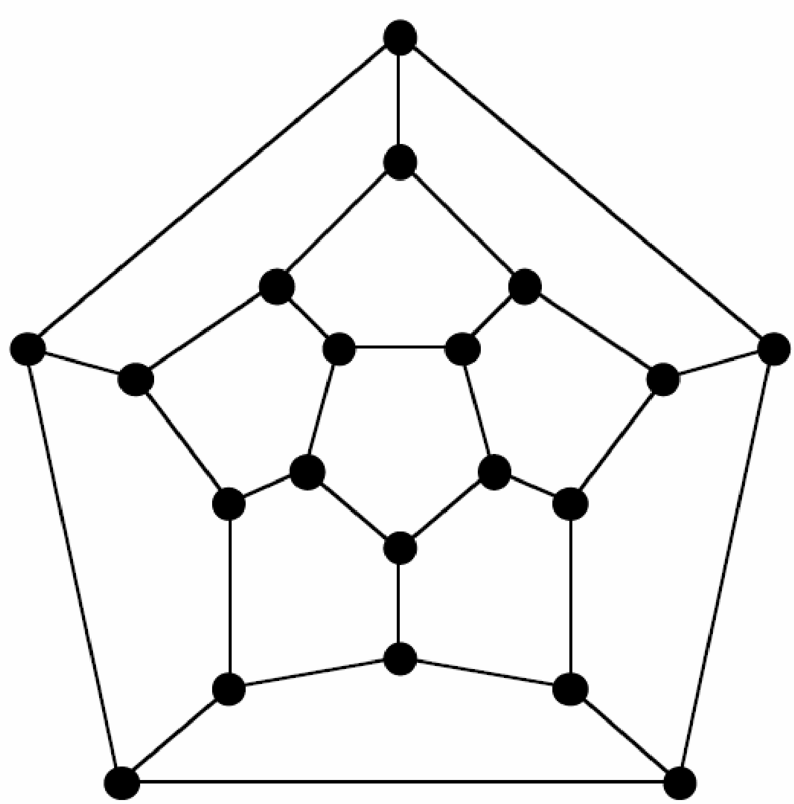
- A connected multigraph has an Euler circuit if and only if each of its vertices has even degree.
- This simple fact can be used to determine if you can “draw a picture without lifting a pencil”.
- A connected multigraph has an Euler path (and no Euler circuit) if there are exactly two vertices of odd degree.
- So what about Konigsburg?
  - Not only is there no Eulerian circuit, there isn’t even an Eulerian path
- Many applications
  - Optimal postman routes, circuit layout, network multicasting, etc.

Hamiltonian paths and cycles

- What if we want every *vertex* to be counted exactly once?
  – Eulerian paths and circuits allow vertices to be counted more than once
- We define a Hamiltonian path $x_0,x_1,\ldots,x_{n-1},x_n$ in G=(V,E) if V = $\{x_0,x_1,\ldots,x_{n-1},x_n\}$ and $x_i \neq x_j$ for $0 \leq I < j \leq n$.
  – A Hamiltonian circuit $x_0,x_1,\ldots,x_{n-1},x_n,x_0$, $n > 1$, if $x_0,x_1,\ldots,x_{n-1},x_n$ is a Hamiltonian path.
- Sadly, there is *no* necessary and sufficient criteria for Hamiltonian circuit
  – There are some theorems with sufficient conditions, and there are a few ways of showing no Hamiltonian circuit.
- Examples: show K_n has a Hamiltonian circuit whenever $n \geq 3$.
  – How about an Eulerian circuit?

Hamilton’s game

- Origin was Hamilton’s Icosian puzzle; wooden dodecahedron (12 regular pentagons) with a peg at each vertex of the dodecahedron, and a string.
  - Each face was a city, i.e., “around the world”.
  - We can visualize the graph as the following…



And more Hamilton…

- (Dirac's Theorem) If G is a simple graph with n vertices, $n \geq 3$, such that the degree of every vertex in G is at least $n/2$, then G has a Hamiltonian circuit.
- (Ore's Theorem) If G is a simple graph with n vertices with $n \geq 3$ such that $\deg(u)+\deg(v) \geq n$ for every pair of nonadjacent vertices u and v in G, then G has a Hamiltonian circuit.
- Sufficient, but not necessary.
  – Graph C_5, for example.

Shortest-path problems

- What if we assign weights to the edges?
- Examples
  - Modeling an airline system: distances, flight time, fares, etc.
  - Modeling a computer network: distance, response times, lease rates
- Such graphs are called **weighted graphs**, and we're interested in the cost (sometimes length) of a path where it's the sum of the weights of the edges of this path.
  - Finding the path of least cost of great interest.

Dijkstra's shortest-path algorithm

- Strategy: first, find the shortest path from start to each of its neighbors. Then, do this repeatedly for each neighbor, but *keep track of the total cost*.
- Initialize a length "function" $L(v_i) := \infty$, $L(a) := 0$, and a set S of seen vertices to the empty set.
- Psuedocode is in the book, but:
  - Add the first unseen vertex of minimum length from the start.
  - Update the lengths in L based on this new vertex's unseen neighbors.
  - Repeat!
- Not only does this provide the shortest path between two vertices in a connected, simple undirected weighted graph, it does so in v^2 time.
- Another algorithm – Floyd – uses Warshall but distances instead of connectivity, and does it for *all* vertices in v^3 time.

Traveling salesman problem

- Goal here is to find the minimum Hamiltonian cycle of a weighted graph
- Imagine the poor salesman who's trying to minimize his travel cost and/or time
- One strategy: brute force
  - Assuming a complete graph, there are (n-1)! different Hamiltonian circuits (counting problem); we need to examine half of these
  - i.e., this is very much an NP problem
- No polynomial algorithm has been found to find this, so people use approximation algorithms that come within some constant ratio of the optimal solution.
  - Within 2% of an exact solution of a 1000-vertex graph in a few minutes of CPU time.

Planar graphs

- Example: consider the problem of joining three houses to each of three separate utilities, without crossing any of the connections.
  - This is a bipartite graph $K_{3,3}$.
  - Another way of phrasing it: can we draw this graph in the plane such that no two links cross?
  - No.
- A graph is called *planar* if it can be drawn in the plane without any edges crossing (where a crossing of edges is the intersection of the lines or arcs representing them at a point other than their common endpoint). Such a drawing is called a *planar representation* of the graph.
  - Note that the graph doesn't have to be drawn without crossed edges for it to be planar; it has to have the *potential* to be drawn without crossed edges.

Several examples

- Is K_4 planar? (yes)
- Is Q_3 planar? (yes)
- So why not $K_{3,3}$?
  - v_1 and v_2 must be both connected to v_4 and v_5; this forms a closed curve that splits the plane, and so on as we add vertices. v_3 to v_4 and v_5 split R_2 (or R_1) into two subregions.
  - No matter where v_6 is placed, it's going to cross a region.
  - Similarly, K_5 is not planar.
- Also important in circuit design (minimizing crossings, if any)

Euler's formula

- Any way to get a better handle on this?
  - Let G be a connected planar simple graph with *e* edges and *v* vertices. Let *r* be the number of regions in a planar representation of G. Then $r = e - v + 2$.
- Can use for the following colloraries.
  - Given a simple, planar G with ≥ 3 vertices, then $e \leq 3v - 6$.
  - If the graph has no circuits of length 3, then $e \leq 2v - 4$.
  - If G is a connected planar simple graph, then G has a vertex of degree not exceeding 5.
- Can we be more precise?

Proving non-planarity?

- We can use the knowledge that $K_{3,3}$ and K_5 are non-planar to help concisely prove any non-planar graph.
- First, we define **homeomorphic graphs** G_1 and G_2 if they can be obtained from the same graph by a sequence of elementary subdivisions.
  - An elementary subdivision is one formed by removing an edge {u,v} and adding w along with edges {u,w} and {w,v}
  - Fundamental idea: adding a vertex doesn't *reduce* the non-planarity of a graph

Kuratowski’s theorem

- A graph is nonplanar if and only if it contains a subgraph homeomorphic to $K_{3,3}$ or K_5.
  – Tip re homeomorphism: if you “smooth” out a node to find a subgraph, you can only do it “one way”.
- Let’s take a look at a couple of examples…
  – Pages 611, 612

Graph Coloring

- Graph coloring has had many contexts and useful applications
- Simple one: map coloring
  - We can convert a map into a graph by using the *dual graph*, which is defined as a graph whose region is represented by a vertex and whose edges beween two vertices signify that those two regions touch.
  - Thus, map coloring reduces to coloring the vertices of a dual graph.
- A *coloring* in a simple graph is the assignment of a color to each vertex of the graph so that no two adjacent vertices are assigned the same color.

Minimize # of colors

- We'd like to figure out the minimum numbers of colors for a graph – the **chromatic number**.
  - The Four Color Theorem says that no planar graph needs more than four colors.
  - Was proven by counterexample; if it was false, one of approximately 2000 different types of graphs would have more than 5 colors, and proved that none of these types existed.
- Nonplanar graphs, of course, can have arbitrarily large chromatic numbers.
- To prove a chromatic number, we must show two things:
  - Show the graph can be colored using *k* colors;
  - Show that it cannot be colored by fewer than *k*.
- Some examples… (page 616)

More examples…

- What's the chromatic number of K_n? What of $K_{m,n}$? (n and 2)
- What's the chromatic number of C_n? (Hint: there are two different answers)
- The best algorithms for finding the chromatic number of a graph have exponential worst-case time complexity.
- Applications
  - Scheduling final exams: vertices represent courses and edge represents common students; colors denote slots
  - Frequency assignments: each vertex is a station, edges represent overlap (e.g., closer than 150 miles), and colors represent the channel assignments.

Next time

- Trees
- Final exam