Lecture 18: 6/2/2009

Announcements: On MyUCD: Ps8, Ps9 solutions, Sample final
FINAL EXAM: NOTE ROOM CHANGE Monday June 8, 10:30-12:30 6 Olson, Open Book, 1 double sided sheet of paper for notes.

OH shifts: Me, 12-1 this Thursday (instead of 1-2)
Nick: 2-4 Friday (after the review session)

Review Session: Friday 12:10-2, 146 Olson

Today: o Students evaluate Me and TA’s // 10 mins

o Graph theory
  o Hints for the final,

Graph theory

1. Notion of a graph

Def: A (finite, simple) Graph G=(V,E) is an ordered pair
  - where V is a nonempty finite set (the "vertices" or "nodes")
  - where E is a collection of two-elements subsets of V (the "edges")
No Parallel edges (at most one edge \{a,b\}). If we allow them G is a multi-graph
No self loops (edge \{a,a\})
G is an Undirected graph:

Graphs are used to represent all sorts of things: we already saw for functions and relations, Finite State Machines, and recursion trees, but also for networks (vertices are computers/routers, edges are communication lines); road networks (cities/intersections, highways/streets); friendships (people, relationship), call graphs (functions, who calls who), many, many more …

Conventional representation: picture.
Be clear: the picture is NOT the graph, it is a representation of the graph.
are the SAME graph.

Def: Two vertices \( v, w \) of a graph \( G=(V,E) \) are **adjacent** if \( \{v,w\} \in E \).

Def: The **degree** of a vertex \( d(v) = |\{v,w\} : w \in V| \)

I like \( \{x,y\} \) for an edge, emphasizing that \( \{x,y\} \) are unordered. Will sometimes see \( xy \) or \( (x,y) \), but both "look" like the order matters, which it does not here (used for directed graphs).

Usually write \( n=|V|, m=|E| \)

2. Paths in graphs

\[ \text{Def: A path } P=(v_1, \ldots, v_n) \text{ in } G=(V,E) \text{ is a sequence of vertices s.t.} \]
\[ \{v_i,v_{i+1}\} \in E \]
\[ \text{for all } i \text{ in } \{1,\ldots, n-1\}. \text{ Note: we exclude the trivial path } a-b-a \text{ that repeats the same edge twice.} \]

A path is said to **contain** the vertices and to **contain** the edges \( \{v_i,v_{i+1}\} \). The **length** of a path is the number of edges on it.

A **cycle** is a path of length 3 or more that starts and stops at the same vertex and includes no repeated vertices apart from the first vertex being the last.

A graph is **acyclic** if it contains no cycle.

A graph \( G=(V,E) \) is **connected** if, for all \( x,y \) in \( V \), there is a path from \( x \) to \( y \).
3. Trees

Def: A _**tree**_ is a connected acyclic graph.
Def: A _**leaf**_ (of a tree) is a vertex of degree one (or zero if G has only one node).

**Picture.**

**Thm1:** Any tree has at least one leaf:

**Proof:** start at some vertex \( v \) in G. If \( \text{deg}(v) = 1 \) or zero, then done, otherwise, let \( w \) be adjacent to \( v \), again if \( w \) is a leaf, done, otherwise it has degree at least 2, so has an adjacent node say \( x \) different from \( v \). Repeat this argument until you get to a leaf. Since there is no cycle you must eventually get to a vertex that has no additional neighbor, and is thus a leaf.

4. Eulerian and Hamiltonian graphs

.............................

Def: A graph G is _**Eulerian**_ if it there is a cycle C in G that goes through every edge exactly once.

A graph G is _**Hamiltonian**_ if there is a cycle that goes through every vertex exactly once.

**Theorem:** (Euler) A connected graph G=(V,E) on \( \geq 3 \) vertices is Eulerian

Iff every vertex of G is of even degree.

**Proof:** \( \rightarrow \) Choose some \( s \). A Graph is Eulerian means there is a path that starts at \( s \) and eventually ends at \( s \), hitting every edge (once). Put a label of 0 on every vertex. Now, follow the path. Every time we enter a vertex or exit a vertex, we increment the label. At end of traversing the graph, label(v) = degree(v) and this is even.

\( \leftarrow \) (sketch) If every vertex is of even degree, at least three vertices. Start at \( s \) and grow a cycle C of unexplored edges until you wind up back at \( s \). You never "get stuck" by even-degree constraint. If every edge explored:Done. Otherwise, find contact point of C and an unexplored edge (exists by connectedness) and grow out from there. Splice together the paths.
So there is a trivial algorithm to decide if $G$ is Eulerian: just check if all its vertices are of even degree.

Amazing fact: There is no efficient algorithm known to decide if a graph is Hamiltonian. Easy to do so using a slow algorithm: try all $n!$ orderings of the vertices. (Most computer scientists believe that no such algorithm exists.)

5. Longest and shortest paths

Def: A _shortest path_ between two vertices $x$ and $y$ is a path from $x$ to $y$ such that there is no shorter (=fewer edges) path from $x$ to $y$. (more general versions put distances on edges and then we want the path with the smallest sum of distances).

A _longest path_ between two vertices $x$ and $y$ is a _simple path_ (=no repeated vertices) from $x$ to $y$.

Claim: There is an efficient algorithm to identify a shortest path between two designated vertices in a graph. (You will learn one in ecs122A and probably ecs60)

Amazing fact: There is no efficient algorithm known to find a longest path from $x$ to $y$. (Most computer scientists believe that no such algorithm exists.)

6. Colorability

.................

**Def:** A graph $G = (V,E)$ is *$k$-colorable* if we can paint the vertices using "colors" $\{1, ..., k\}$ such that no adjacent vertices have the same color.

**Def:** A graph is bipartite if it is 2-colorable. In other words, we can partition $V$ into $(V_1, V_2)$ such that all edges go between a vertex in $V_1$ and a vertex in $V_2$.

**Proposition:** There is a simple and efficient algorithm to decide if a graph $G$ is 2-colorable. **Proofs:** Modify DFS. Or show ad hoc algorithm directly...

Amazing fact: There is no reasonable algorithm known to decide if a graph is 3-colorable.

(Most computer scientists believe that no such algorithm exists.)