Today:

- Graph Theory, continued

**Graph theory**

1. Review of definitions and vocabulary
3. Paths, cycles, Eulerian graphs

---

### 1. Basic Definitions

**Def:** A (finite, simple) graph $G=(V, E)$ is an ordered pair
- $V$ is a finite nonempty set (the vertices or nodes)
- $E$ is a set of two-elements subsets of $V$ (the edges)

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, in a simple graph, it does not.

Usually people use $n=|V|$ and $m=|E|$; alternatively, $v=|V|$ and $e=|E|$ looks nice and suggestive.

There are many other “kinds” of graphs—for example, in a directed graph (digraph), the edges (now often called arcs) are ordered pairs, instead of unordered pairs. We sometimes allow graphs with self-loops (and edge between a vertex and itself) or multiple edges (two or more different edges connecting a pair of nodes). In a network, each edge (or arcs) has a real-valued weight. People consider infinite graphs. We even have graphs where an even can be incident (touch) touch than two vertices (hypergraphs). None of these variants are not allowed in simple graphs. For this lecture, we’re going to stick to them.

Conventional representation: a picture. (Draw some.) But be clear: the picture is NOT the graph, it is a representation of the graph. The graph is the pair $(V, E)$.

Some “special” graphs – a clique of size $n$, $K_n$, and complete bipartite graphs on $n$ “boys” and $m$ “girls”, $K_{n,m}$.

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

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

**Def:** The neighbor set of $v$ in a graph $G=(V, E)$ is $N(v) = \{w \in V: \{v, w\} \in E\}$. 

Note that $\deg(v) = |N(v)|$.

Some counting

Question: How many different graphs are there on $V=\{1,\ldots,n\}$? $2^{\binom{n}{2}} = 2^{n(n-1)/2}$

Question: What is the maximal and minimal degrees of an $n$-vertex graph?

Question: Count how many edges in $K_n$ and $K_{n,m}$.

Isomorphism

We don’t usually care about the names of points in $V$, only how they’re connected up. If two graphs are the same, up to renaming, we call them isomorphic. Formally, graphs $G=(V,E)$ and $G’=(V’,E’)$ are isomorphic if there is a permutation $\pi: V \rightarrow V’$ such that $\{v,w\} \in E$ iff $\{\pi(v),\pi(w)\} \in E’$. The properties of graphs that matter are those that are invariant under isomorphism.

Def: Graphs $G=(V,E)$ and $G’=(V’,E’)$ are isomorphic if there exists a permutation $\pi$ such that $\{x,y\} \in E$ iff $\{\pi(x),\pi(y)\} \in E’$.

Proposition: Isomorphism is an equivalence relation.

Amazing fact: there is no efficient algorithm known to decide if two graphs are isomorphic. (Most computer scientists believe that no such algorithm exists.) One of the biggest open questions in computer science.

Maybe show how to prove two graphs are non-isomorphic using an interactive proof. [Didn’t do this]

Prop: $\Sigma_\nu \deg(\nu) = 2m$
Bipartite graphs

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. Formally,

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: A graph is bipartite iff it is 2-colorable iff it has no odd-length cycles, which can be done after we introduct that vocabulary.)

Proposition: There is a simple and efficient algorithm to decide if a graph $G$ is 2-colorable / bipartite.
Proof: Modify DFS.

Initially, all vertices are uncolored: color[v]=UNCOLORED
While there are uncolored vertices $v$ in $G$ do DFS($v$,0)

Algorithm DFS($v$,b)
  color[$v$] = $b$
  for each uncolored $w$ in $N(v)$ do DFS($w$, 1-$b$)

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.)

Proposition [Appel, Haken 1989] Every planar graph is 4-colorable.

3. Paths, Cycles, connectivity, and Eulerian cycles

Def: A path $p=(v_1, ..., v_n)$ in $G = (V,E)$ is a sequence of vertices s.t. $\{v_i,v_{i+1}\} \in E$
for all $i$ in \{1,..., n-1\}.

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 three or more that starts and ends at the same vertex and includes no repeated edges.

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 $y$.

The components of a graph are the maximal connected subgraphs.
(Graph $G'=(V',E')$ is a subgraph of graph $G=(V,E)$ if $V' \subseteq V$ and $E' \subseteq E$.)

Alternative definition of components: Say that $x \sim y$ (these vertices are in the same component) if there is a path from $x$ to $y$. Prop: this is an equivalence relation. Its blocks (equivalence classes) are the components.
Alternative definition of a component: the component containing \( v \) is all vertices connected to \( v \) by paths of any lengths; and all the induced edges (the edges of the original graph that span vertices in the component).

Describe an algorithm, based on DFS, for counting the number of components of a graph and identifying them.

**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 \( n \geq 3 \) vertices is Eulerian if

\[
\text{every vertex of } G \text{ is of even degree.}
\]

**Proof:**

\( \Rightarrow \) Choose \( s \). Graph is Eulerian mean there is a path that starts at \( s \) and eventually ends at \( s \), hitting every edge. Put a label of 0 on every vertex. Now, follow the path. Every time we exit a vertex, increment the label. Every time we enter a vertex, increment the label. At end of traversing the graph, label(\( v \)) = deg(\( v \)) 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 “reasonable” algorithm known to decide if a graph is Hamiltonian.

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