T minus 3 classes

- Homework 9 is out
  - Due in a week
- Extra Credit on Piazza
- No class Thursday
  - (same for office hours)
- Email me if you have any questions

CMSC 203: Lecture 24

Graphs

Graphs

- Discrete structures consisting of **vertices** and **edges**
- Multiple kinds
  - Direction for edges?
  - Multiple edges for each vertex pair?
  - Are loops allowed?
  - Do edges have weights?
- Can solve problems in almost any field

Uses of Graphs

- Efficient routes for walking / trucks
- Map coloring problems
- Implementing circuit on planar circuit board
- Distinguish between similar chemical compounds
- Determine if two computers are connected
- Schedule exams / television stations

Definition of a Graph

- Graph $G = (V, E)$
  - V: vertices (nodes)
    - Nonempty, possibly infinite
  - E: edges
    - One or two vertices per edge (endpoints)

Types of Graphs

- **Simple graph**
  - Each edge connects two different vertices
  - No two edges connect to the same pair of vertices
- **Multigraph**
  - Multiple edges connect same vertices
  - Useful for representing *multiplicity*
- **Psuedograph**
  - May contain loops and multiple edges
  - Useful for representing self-affecting vertices

Types of Graphs

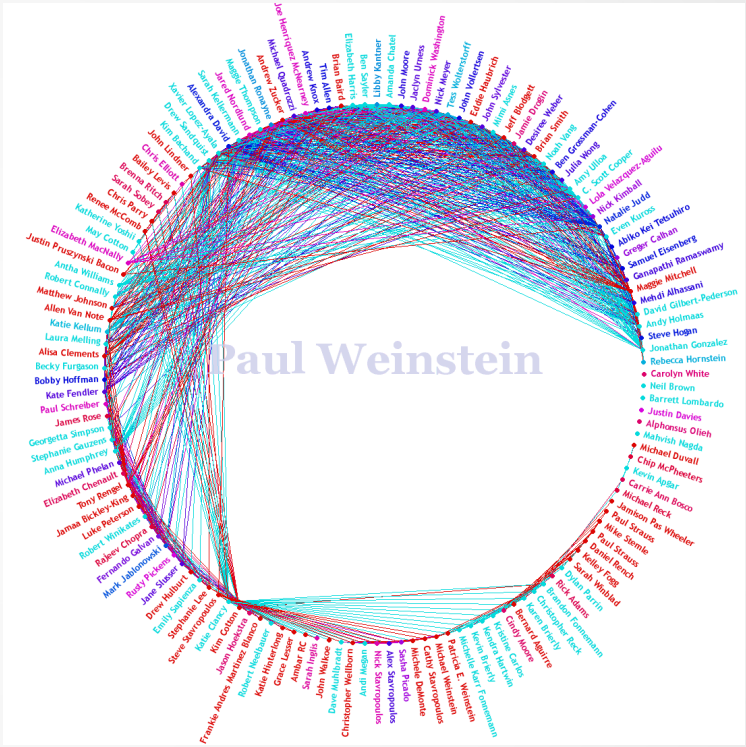
- **Undirected graph**
  - No directions on edges
  - Connections work both directions
- **Directed graph** (digraph)
  - Edges have a particular direction (we've seen this)
- **Mixed graph**
  - May contain a mix of directed and undirected edges

What To Know

- Three key questions when looking at graphs:
  - Are edges of graphs directed or undirected (or both)?
  - Are multiple edges present?
  - Are loops present?

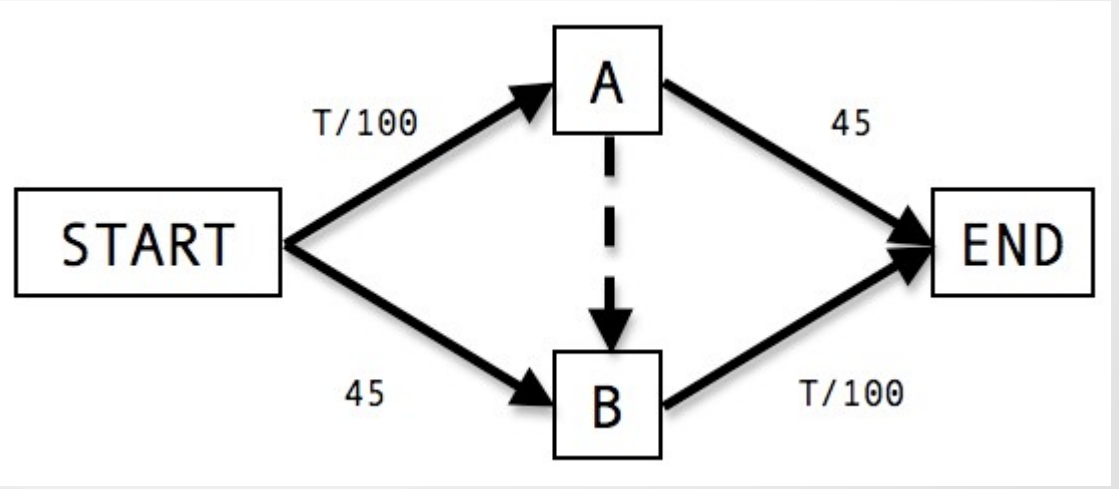
Graph Models

- Social networks
  - Acquaintanceship graph
  - Influence Graph
  - Collaboration graph
- Communication networks
  - Call Graphs
- Information Networks
  - Web Graph
  - Citation Graph



Graph Models (cont)

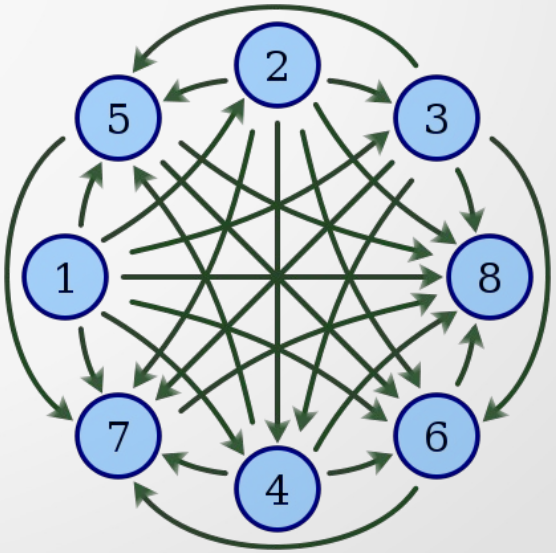
- Software Design Applications
  - Module Dependency Graphs
  - Precedence Graphs / Concurrent Processing
- Transportation network
  - Airline Routes
  - Road Networks



Braess's paradox

Graph Models (cont)

- Biological Networks
  - Niche Overlap Graphs in Ecology
  - Protein Interaction Graph
- Tournaments
  - Round-Robin Tournaments
  - Single-Elimination Tournaments



Definitions for **Undirected** Graphs

- Two vertices *u* and *v* are **adjacent** (or **neighbors**) if *u* and *v* are endpoints of an edge *e* (*incident with v*)
- Set of all neighbors of *v* is called the **neighborhood** of *v*
  - Denoted *N(v)*
- **Degree** of a vertex is number of edges incident with it
  - Except, a loop contributes twice to the degree
  - Denoted by deg(*v*)
- A vertex with degree zero is **isolated**
- A vertex with degree one is **pendant**

Handshaking Theorem

- Let G = (V, E) with *m* edges (in undirected graph)

$$2m = \sum_{v \in V} deg(v)$$

- An undirected graph has an even number of vertices of odd degree

Definitions for **Directed** Graphs

- If (u, v) is an edge in a graph, u is said to be **adjacent to** v, and v is said to be **adjacent from** u
- The **in-degree of a vertex** v is the number of edges with v as their terminal vertex
  - Denoted by deg-(v)

$$\sum_{v \in V} deg^-(v) = \sum_{v \in V} deg^+(v) = |E|$$

- The **out-degree of a vertex** v is the number of edges with v as their initial vertex
  - Denoted by deg+(v)
- A loop contributes 1 to both in-degree and out-degree

Special Graphs

- Complete Graph: K_n
  - Simple graph that contains one edge between each pair of distinct vertices
- Cycles: C_n
  - Consists of *n* vertices $v_1, v_2, \ldots, v_n$ and edges $\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{n-1}, v_n\}, \text{ and } \{v_n, v_1\}$
- Wheel: W_n
  - Add additional vertex to C_n and connect vertex to each of the n vertices

Bipartite Graphs

- A simple graph is bipartite if its vertices can be partitioned into two disjoint sets such that every edge connects a vertex in both sets
  - No edge connects vertices in the same set
- Can be verified if and only if it is possible to assign one of two different colors to each vertex of the graph so no two adjacent vertices are assigned the same color
  - Known as **Graph coloring**
- Useful for applications that involve matching elements between sets