Lecture 16: Graphs

CS 111: Intro to Computational Science
Spring 2023

Ziad Matni, Ph.D.
Dept. of Computer Science, UCSB

Administrative

- New homework due **Tuesday**
- Lab tomorrow
- Quiz 5 grades are on Canvas
- Remember: NO CLASS NEXT MONDAY!
- Your last quiz is next week Wednesday (Quiz 7)
  - Will be on lectures 15, 16 (PCA, Graphs)
- Re: Final Exam
  - **Wed. June 14th** from **9 – 11 am** → **ARRIVE 10 mins EARLY!**

Networks Can Be About… Anything

Network-centric views help us look at things from a certain perspective.

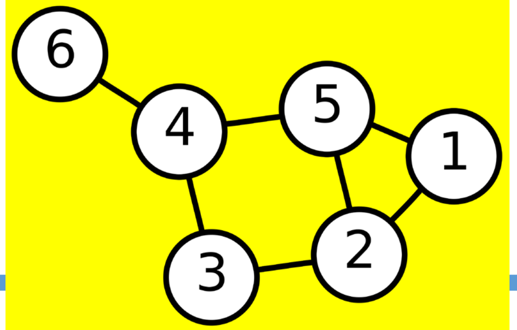
Example: **Network of Ideas**

Check out this short talk!

http://www.ted.com/talks/eric_berlow_and_sean_gourley_mapping_ideas_worth_spreading

(7:51)

Graph Theory

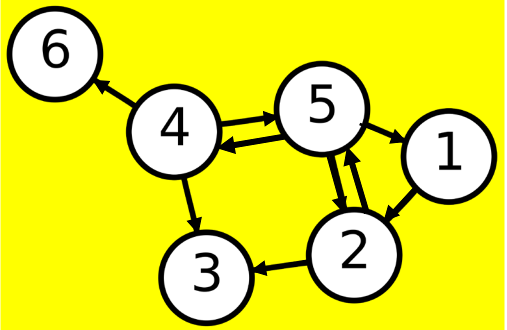


Undirected graph

- Graphs: mathematical structures that model *pairwise relations* between objects

- Made up of **vertices** (aka **nodes**) which are connected by **edges** (aka **links**).

- Some graphs are undirected (use straight lines to symbolize edges)
  - where edges link two vertices symmetrically (A → B means B → A too)



Directed graph

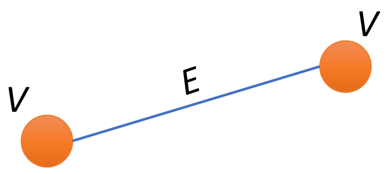
- Some graphs are directed (use arrows to symbolize edges)
  - where edges link two vertices asymmetrically (A → B ≠ B → A)

Studying Networks of… *anything…*

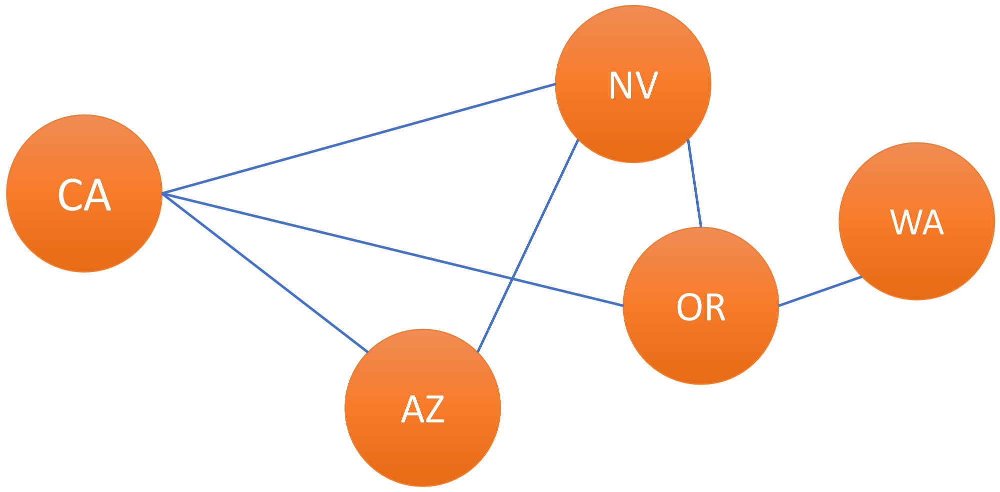
- Based on Graph Theory

- Network structures can describe a lot of phenomena
  - How certain cancer cell form
  - How people make social links with one another (i.e. social networks)
  - How words and their meanings are linked together (in linguistics)
  - *How documents end up being linked on the WWW*
  - *How internet protocols works*

Vertices and Edges *aka* Nodes and Links



- *The only 2 components of any graph/network!*
- Vertices (nodes/entities) are connected to each other via edges (links/relationships)
- **Vertices** can be: Web docs, people, words, *etc*…
- **Edges** can be: “has URL link to”, “knows”, “found before (other words)”, *etc*…

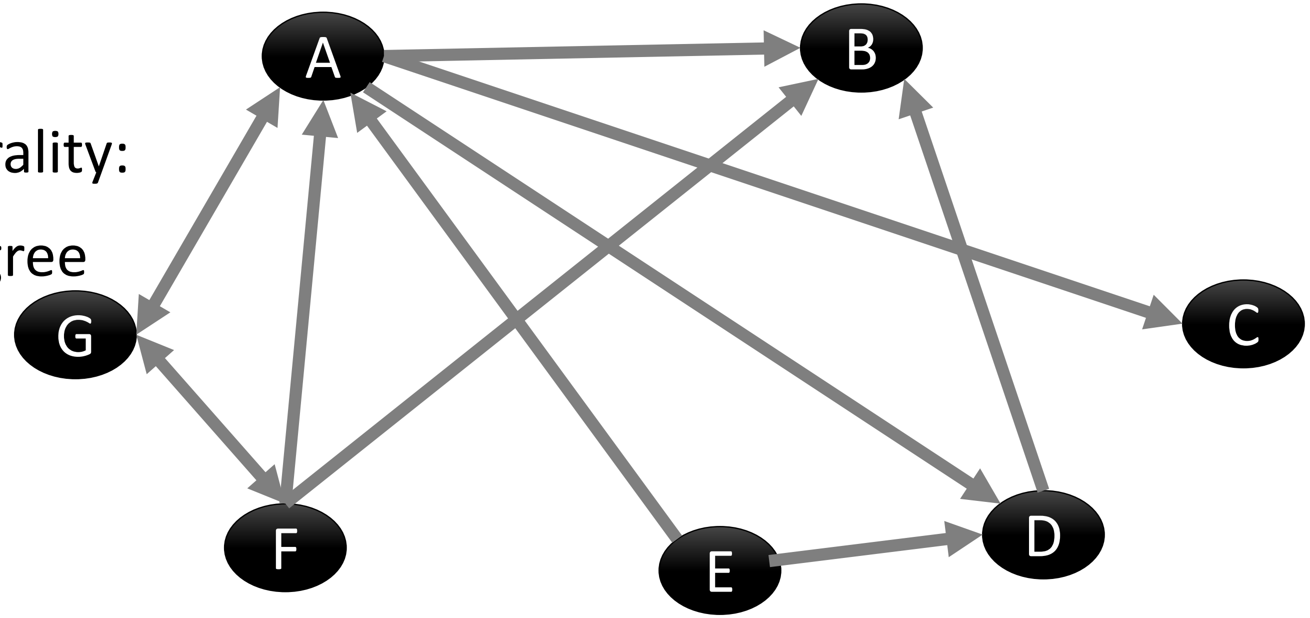


Degree Centrality

- Measures of the "Importance" of a Vertex/Node
  - Simply this: Importance is based only on how many edges/links does a node have attached to it?

- Example: Looking at Node A's degree centrality:

  3 in-degree, 4 out-degree



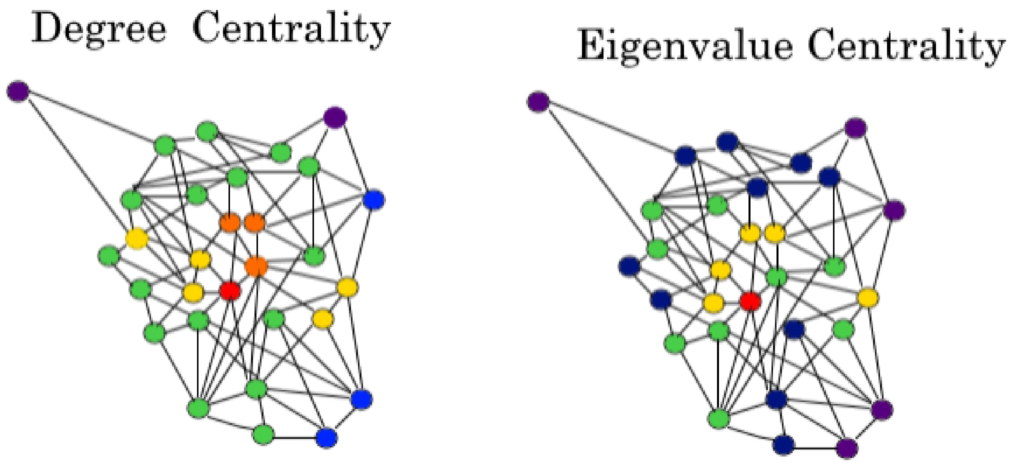
- **Pros**: It's a simple measure of node "importance" or "connectivity"

- **Cons**: It's too simple a measure…

Eigenvector Centrality

- A more sophisticated version of degree centrality
  - Not all "highly central" nodes have the same "importance"
  - *Main Idea: A node is **important** if it is linked to **other important** nodes*
- Like *degree centrality*, this one calculates number of links on a given node
  - **BUT** then each adjacent node is **weighted** by its *own* centrality
  - So, a link to a high-centrality node **"counts for"** more than one going to a less-connected node

Eigenvector Centrality



- Popular in measuring ***influence*** and ***access*** to influential nodes in a network

- The Google search algorithm (**PageRank**) uses a *variation on this measure* to decide how "important" a webpage is

Other Centrality Measures

- Example: "**Betweenness Centrality**"

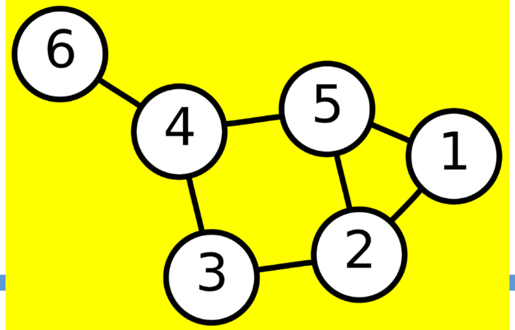
- If a node has *high* betweenness centrality

  ➔ It has a *large potential for controlling "flows"*

  (like messages, influence, resources) through the network

  because it "shortens" the "number of hops" between any other 2 nodes

The Degree Matrix



- Network graphs can be described using matrices
- **Degree matrix**
  - An **n**x**n diagonal** matrix: describes the **degree** of each node in a graph with ***n*** nodes
  - Usually used only for <u>undirected</u> graphs
  - For directed graphs, you need one matrix for *in-degrees* and another for *out-degrees*
  - **D** =

$$\begin{pmatrix} \mathbf{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{3} & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{3} & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{3} & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{1} \end{pmatrix}$$

Node 1 has degree = 2

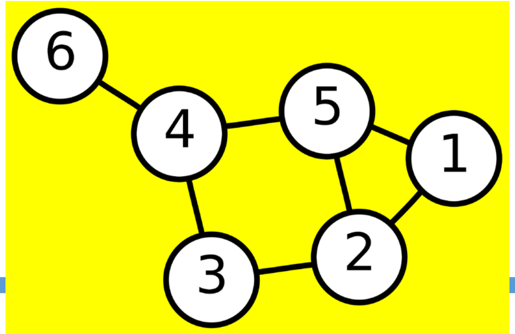
Node 2 has degree = 3

Note, that **trace(D)**, i.e. the sum of diagonals, is ***twice*** the number of edges for obvious reasons.

i.e. **trace(D) = *2m***, where ***m*** = no. of edges

In the example: m = 7, trace(D) = 14

The Adjacency Matrix



- Shows what vertices (nodes) are connected in an ***n*** sized graph
  - Is always an **n$_x$n** matrix
  - In a large graph, it is typically a sparse matrix

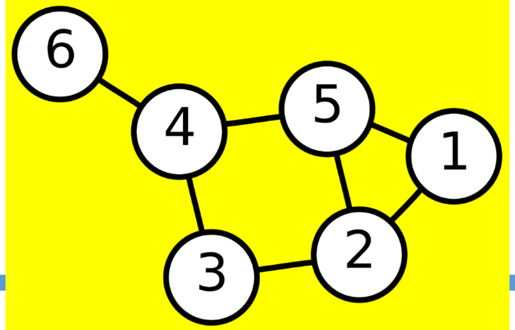
- A very compact way of representing a graph
  - Only need **1** (for ‘connected’) and **0** (for ‘not connected’) values in it
  - Also, we can easily derive **D** from **A** *(how?)*

- In an ***undirected*** graph, **A** is always **symmetrical** (if node x →y, then node y → x too)
- In a ***directed*** graph, **A** is *not necessarily* symmetrical

$$
\begin{array}{ll}
 & \text{Nodes: } \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 \end{matrix} \\
\mathbf{A} = \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} & \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}
\end{array}
$$

Note: This shows an undirected graph

The Graph Laplacian Matrix



- Shows BOTH **degree** and **adjacency** information in an ***n*** sized graph
  - Calculated as: **L** = **D** – **A**
  - Is always an **nxn** matrix

- **L** = $\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$

L has interesting properties that combine properties for **D** and **A** and other things…!

Graph Laplacian Characteristics

$$\begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$$

- Similar to **A**, vertex connections are spelled-out (here, as all the **-1** entries)
- Similar to **D**, the trace of **L** = $2m$; where m = number of edges
- **In a GL matrix, every row adds to zero, so does every column**
- Let vector $\mathbf{v_0} = [1,1,1,1,1]^T$; Note that: $\mathbf{L\, v_0 = \lambda_0\, v_0 = 0}$

$$\begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

  - Therefore, λ_0 of **L** is always zero!
  - Therefore, **L** is a singular matrix (in the example given, the rank of this 6x6 matrix is **5**)

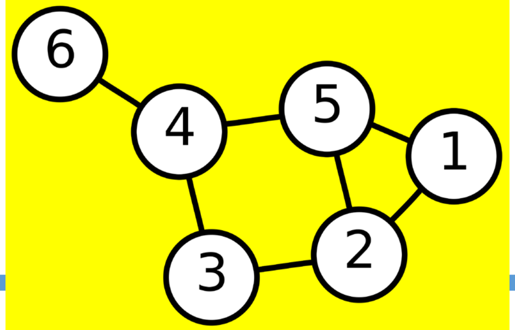
- L is **symmetric** and **positive semi-definite** (i.e. it's eigenvalues are ≥ 0)

Graph Laplacian Matrix Uses

- Clustering algorithms
  - Applied in Machine Learning, task scheduling in multiprocessor computers, electronic integrated circuit design, etc…

- Kirchkoff's Theorm and Spanning Trees in Math/Theoretical C.S.
  - Spanning tree = a path in a network that includes all the nodes
  - Number of Spanning trees in a network $= \frac{1}{n}(\lambda_1 \lambda_2 \ldots \lambda_{n-1})$
    - where λ_i are non-zero eigenvalues of the GL of the network
  - Helps to reduce a graph to a tree

The “Density” of a Graph/Network



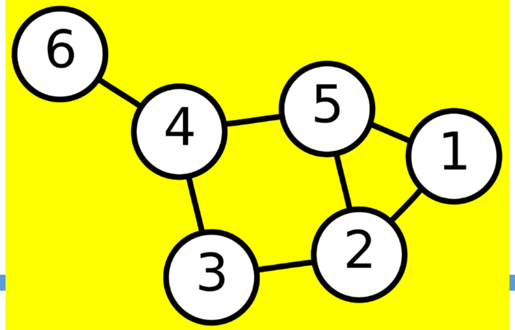
- Tells us how many links/edges are “realized” in the network.

- A *fully-connected network* (den. = 100%) is where *every* node/vertex connects to *every other* node

- *Thought exercise: If I have **n** vertices, how many connections can I make?*

- The ***max*** number of edges in a network with **n** nodes is: **½ n(n-1)**
  - Assumes an undirected graph/network
  - **Double** that number for directed graph (i.e. it’s **n(n-1)**)

The “Density” of a Graph/Network



- Finally, the density of a network (**d**) is defined as:

d = (number of actual edges) / (max. number of edges)

- Example: in the above network, the density is:

$$d = 7 / (½ * 6 * 5) = 7 / 15 = \underline{46.7\%}$$

- Density is a useful measure when studying the efficiency of how stuff disperses in a network
  - The most efficient networks for dispersion tend to have “medium” densities…!

Your TO DOs!

- Finish new assignment by Tuesday
- Lab tomorrow!

</LECTURE>