Block 2: Intro to graphs

ELEC 573: Network Science and Analytics

Santiago Segarra

Electrical and Computer Engineering
Rice University
segarra@rice.edu

Fall 2021

You are here

Wk.	Date	Topic	HW	Project
1	23-Aug	Introduction to course	HW0 out	
2	30-Aug	Graph theory / Centrality measures	HW0 solutions posted	
3	6-Sep	LABOR DAY (no class)	HW1 out	
4	13-Sep	Centrality measures / Community detection		
5	20-Sep	Community detection		
6	27-Sep	Signal Processing and Deep learning for graphs	HW1 due	
7	4-Oct	Signal Processing and Deep learning for graphs	HW2 out	
8	11-Oct	FALL BREAK (no class)		
9	18-Oct	Network models	HW2 due	
10	25-Oct	Network models	HW3 out	Project proposal due
11	1-Nov	Epidemics		
12	8-Nov	Inference of network topologies, features, and processes	HW3 due	
13	15-Nov	Inference of network topologies, features, and processes		
14	22-Nov	Inference of network topologies, features, and processes		Project progress report
15	29-Nov	Inference of network topologies, features, and processes		
	13-Dec	Project presentation (video recording) and final report due		

Homework, project and grading

(I) 3 homework sets worth 40% (plus an ungraded Homework 0)

- Mix of analytical problems and programming assignments
- Collaboration accepted, welcomed, and encouraged
- However, the submitted work must be your own

Homework, project and grading

(I) 3 homework sets worth 40% (plus an ungraded Homework 0)

- Mix of analytical problems and programming assignments
- Collaboration accepted, welcomed, and encouraged
- However, the submitted work must be your own

(II) Research project on a topic of your choice, worth 60%

- Important and demanding part of this class. Three deliverables:
  1) Proposal by the end of week 9, worth 10%
  2) Progress report by the end of week 12, worth 15%
  3) Final report and recorded presentation, worth 35%

Homework, project and grading

(I) 3 homework sets worth 40% (plus an ungraded Homework 0)

- Mix of analytical problems and programming assignments
- Collaboration accepted, welcomed, and encouraged
- However, the submitted work must be your own

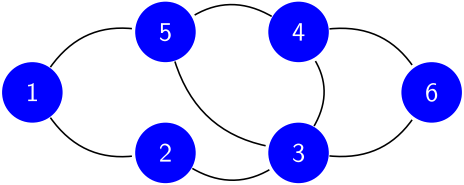
(II) Research project on a topic of your choice, worth 60%

- Important and demanding part of this class. Three deliverables:
  1) Proposal by the end of week 9, worth 10%
  2) Progress report by the end of week 12, worth 15%
  3) Final report and recorded presentation, worth 35%
- This is a research-oriented graduate level class

  $\Rightarrow$ Focus should be on thinking, reading, asking, implementing

A few things about handing-in homework

- All submissions must be via Canvas
- Can be scanned copies of handwritten work if you are tidy
- All homework is released on Monday evenings
- All homework is due on Tuesday by midnight
- For coding exercises, ready-to-run code must be included
  - $\Rightarrow$ Submit all your work in a compressed folder
  - $\Rightarrow$ Jupyter notebooks highly appreciated
  - $\Rightarrow$ 'lastname_homework_i'
- If anything comes up, please come talk to me with time!
  - $\Rightarrow$ An ounce of prevention is worth a pound of cure



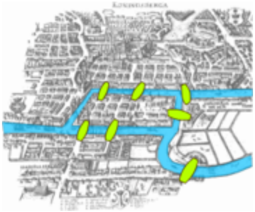
- Graph $G(V,E)$ $\Rightarrow$ A set V of vertices or nodes
  $\Rightarrow$ Connected by a set E of edges or links
  $\Rightarrow$ Elements of E are unordered pairs (u,v), $u,v \in V$
- In figure $\Rightarrow$ Vertices are $V = \{1,2,3,4,5,6\}$
  $\Rightarrow$ Edges $E = \{(1,2),(1,5),(2,3),(3,4), \ldots$
  $(3,5),(3,6),(4,5),(4,6)\}$

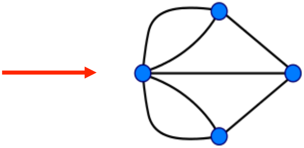
From networks to graphs

- Networks are complex systems of inter-connected components
- Graphs are mathematical representations of these systems
  - $\Rightarrow$ Formal language we use to talk about networks

From networks to graphs

- Networks are complex systems of inter-connected components
- Graphs are mathematical representations of these systems
  - $\Rightarrow$ Formal language we use to talk about networks





- Components: nodes, vertices $\qquad V$
- Inter-connections: links, edges $\qquad E$
- Systems: networks, graphs $\qquad G(V, E)$

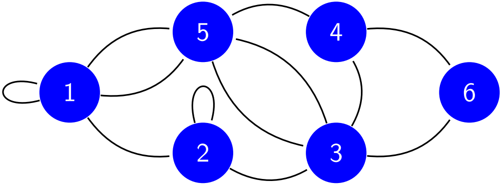
Vertices and edges in networks

Network	**Vertex**	**Edge**
Internet	Computer/router	Cable or wireless link
Metabolic network	Metabolite	Metabolic reaction
WWW	Web page	Hyperlink
Food web	Species	Predation
Gene-regulatory network	Gene	Regulation of expression
Friendship network	Person	Friendship or acquaintance
Power grid	Substation	Transmission line
Affiliation network	Person and club	Membership
Protein interaction	Protein	Physical interaction
Citation network	Article/patent	Citation
Neural network	Neuron	Synapse
$\vdots$	$\vdots$	$\vdots$

Simple and multi-graphs

- In general, graphs may have self-loops and multi-edges

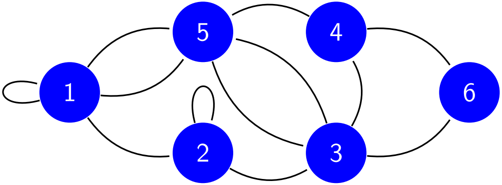
  $\Rightarrow$ A graph with either is called a multi-graph



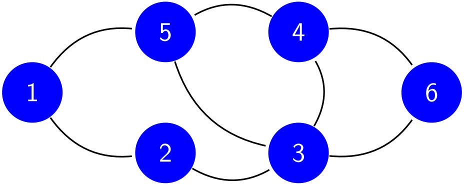
Simple and multi-graphs

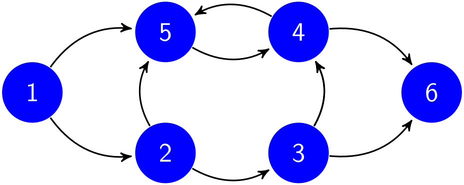
- In general, graphs may have self-loops and multi-edges

  $\Rightarrow$ A graph with either is called a multi-graph



- Mostly work with simple graphs, with no self-loops or multi-edges

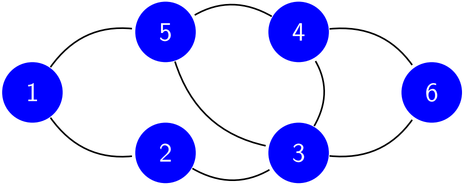




- In directed graphs, elements of E are ordered pairs (u, v), $u, v \in V$
  - $\Rightarrow$ Means (u, v) distinct from (v, u)
- Directed graphs often called digraphs
  - $\Rightarrow$ By convention (u, v) points to v
  - $\Rightarrow$ If both $\{(u, v), (v, u)\} \subseteq E$, the edges are said to be mutual
- Ex: who-calls-whom phone networks, Twitter follower networks

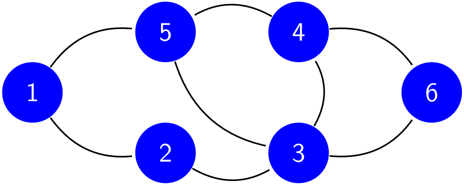
Subgraphs

- Consider a given graph $G(V, E)$

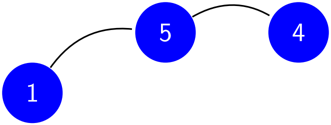


- **Def:** Graph $G'(V', E')$ is an induced subgraph of G if $V' \subseteq V$ and $E' \subseteq E$ is the collection of edges in G among that subset of vertices

- Consider a given graph $G(V, E)$



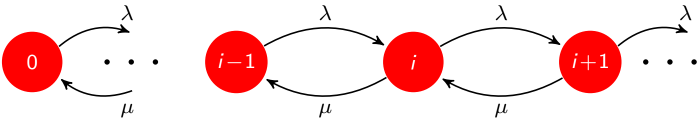
- **Def:** Graph $G'(V', E')$ is an induced subgraph of G if $V' \subseteq V$ and $E' \subseteq E$ is the collection of edges in G among that subset of vertices
- Ex: Graph induced by $V' = \{1, 4, 5\}$



Weighted graphs

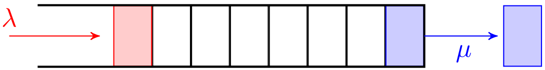
- Oftentimes one labels vertices, edges or both with numerical values

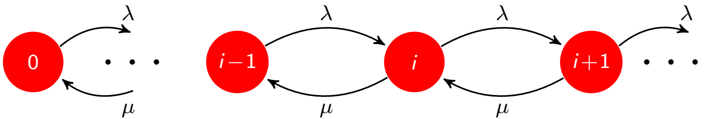
  $\Rightarrow$ Such graphs are called weighted graphs
- Useful in modeling e.g., Markov chain transition diagrams
- Ex: Single server queuing system (M/M/1 queue)



Weighted graphs

- Oftentimes one labels vertices, edges or both with numerical values
  - $\Rightarrow$ Such graphs are called weighted graphs
- Useful in modeling e.g., Markov chain transition diagrams
- Ex: Single server queuing system (M/M/1 queue)





- Labels could correspond to measurements of network processes
- Ex: Node is infected or not with influenza, IP traffic carried by a link

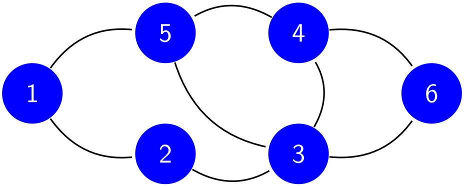
Typical network representations

Network	Graph representation
WWW	Directed multi-graph (with loops), unweighted
Facebook friendships	Undirected, unweighted
Citation network	Directed, unweighted, acyclic
Collaboration network	Undirected, unweighted
Mobile phone calls	Directed, weighted
Protein interaction	Undirected multi-graph (with loops), unweighted
⋮	⋮

- Note that multi-edges are often encoded as edge weights (counts)

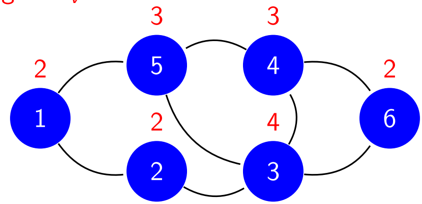
Adjacency

- Useful to develop a language to discuss the connectivity of a graph
- A simple and local notion is that of adjacency
  - $\Rightarrow$ Vertices $u, v \in V$ are said adjacent if joined by an edge in E
  - $\Rightarrow$ Edges $e_1, e_2 \in E$ are adjacent if they share an endpoint in V



- In figure $\Rightarrow$ Vertices 1 and 5 are adjacent; 2 and 4 are not
  $\Rightarrow$ Edge $(1, 2)$ is adjacent to $(1, 5)$, but not to $(4, 6)$

- An edge (u, v) is incident with the vertices u and v
- **Def:** The degree d_v of vertex v is its number of incident edges



- In figure $\Rightarrow$ Vertex degrees shown in red, e.g., $d_1 = 2$ and $d_5 = 3$
- High-degree vertices likely influential, central, prominent.
- The neighborhood $\mathcal{N}_i$ of a node i is the set of all its adjacent nodes

  $\Rightarrow \mathcal{N}_5 = \{1, 3, 4\}$ $\Rightarrow$ In general, $|\mathcal{N}_i| = d_i$

Properties and observations about degrees

- Degree values range from 0 to $|V| - 1$
- The sum of the degree sequence is twice the size of the graph

$$\sum_{v=1}^{|V|} d_v = 2|E|$$

Properties and observations about degrees

- Degree values range from 0 to $|V| - 1$
- The sum of the degree sequence is twice the size of the graph

$$\sum_{v=1}^{|V|} d_v = 2|E|$$

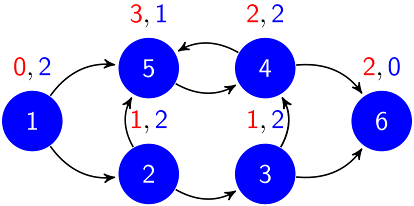
$\Rightarrow$ The number of vertices with odd degree is even

- Degree values range from 0 to $|V| - 1$
- The sum of the degree sequence is twice the size of the graph

$$\sum_{v=1}^{|V|} d_v = 2|E|$$

$\Rightarrow$ The number of vertices with odd degree is even

- In digraphs, we have vertex in-degree d_v^{in} and out-degree d_v^{out}



- In figure $\Rightarrow$ Vertex in-degrees shown in red, out-degrees in blue
  $\Rightarrow$ For example, $d_1^{in} = 0, d_1^{out} = 2$ and $d_5^{in} = 3, d_5^{out} = 1$

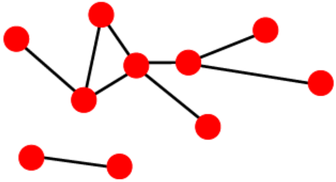
- Let $N(d)$ denote the number of vertices with degree d

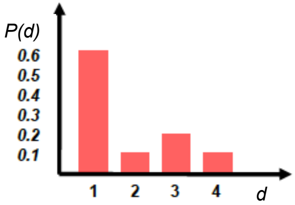
  $\Rightarrow$ Fraction of vertices with degree d is $\text{P}[d] := \frac{N(d)}{|V|}$

- Let $N(d)$ denote the number of vertices with degree d

  $\Rightarrow$ Fraction of vertices with degree d is $\mathrm{P}\left[d\right] := \frac{N(d)}{|V|}$

- **Def:** The collection $\{\mathrm{P}\left[d\right]\}_{d \geq 0}$ is the degree distribution of G
  - Histogram formed from the degree sequence (bins of size one)





- $\mathrm{P}\left[d\right]$ = probability that randomly chosen node has degree d

  $\Rightarrow$ Summarizes the local connectivity in the network graph

Movement in a graph

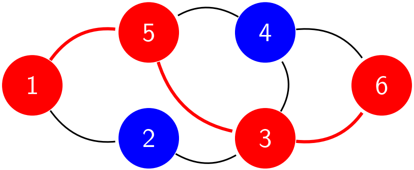
- A path of length l from v_0 to v_l is a sequence

$$\{v_0, v_1, \ldots, v_{l-1}, v_l\}, \text{ where } v_i \text{ and } v_{i+1} \text{ are adjacent}$$

- A path of length l from v_0 to v_l is a sequence

$$\{v_0, v_1, \ldots, v_{l-1}, v_l\}, \text{ where } v_i \text{ and } v_{i+1} \text{ are adjacent}$$

- A simple path is a path without repeated nodes



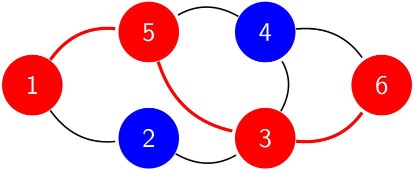
- A closed path ($v_0 = v_l$) is denominated a circuit

  $\Rightarrow$ If no other nodes are repeated, then it is a cycle

- A path of length l from v_0 to v_l is a sequence

$$\{v_0, v_1, \ldots, v_{l-1}, v_l\}, \text{ where } v_i \text{ and } v_{i+1} \text{ are adjacent}$$

- A simple path is a path without repeated nodes



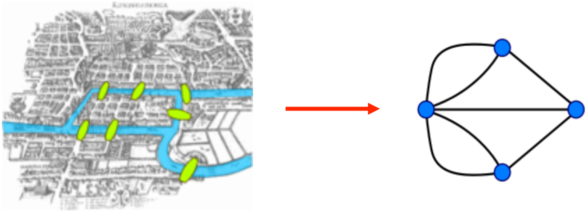
- A closed path ($v_0 = v_l$) is denominated a circuit

  $\Rightarrow$ If no other nodes are repeated, then it is a cycle

- All these notions generalize naturally to directed graphs

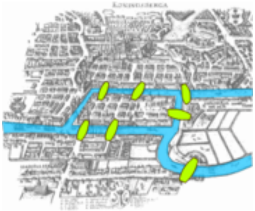
Solution to the bridges of Königsberg

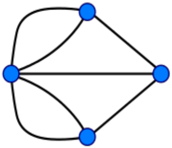
- Can you cross each bridge exactly once in a path?



Solution to the bridges of Königsberg

- Can you cross each bridge exactly once in a path?





- Graph matters, not physical properties of the island
- Suppose a walk existed, what would this imply about the degrees?
- Can such a walk exist in Königsberg?

Distances in a graph

- Assume that the edge weights represent length of walk

  $\Rightarrow$ Length of a path $\Rightarrow$ Sum weights of traversed edges

- The distance between two nodes i and j is the length of the shortest path linking i and j

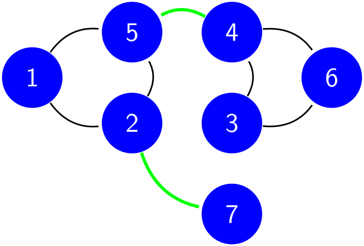
  $\Rightarrow$ In the absence of such a path, the distance is ∞

  $\Rightarrow$ The diameter of the graph is the value of the largest distance

- There exist efficient algorithms to compute distances in graphs

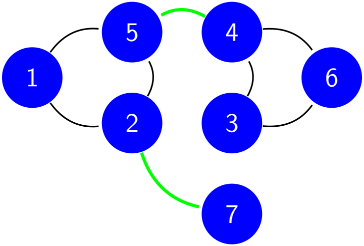
  $\Rightarrow$ Dijkstra, Floyd-Warshall, Johnson, ...

- Vertex v is reachable from u if there exists a $u - v$ path
- **Def:** Graph is connected if every vertex is reachable from every other



Connectivity

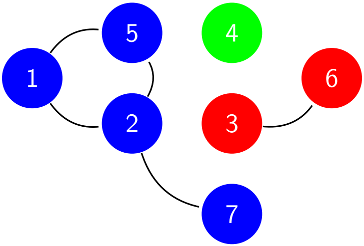
- Vertex v is reachable from u if there exists a $u - v$ path
- **Def:** Graph is connected if every vertex is reachable from every other



- If bridge edges are removed, the graph becomes disconnected

Connected components

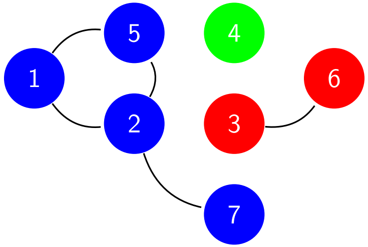
- **Def:** A component is a maximally connected subgraph
  $\Rightarrow$ Maximal means adding a vertex will ruin connectivity



- In figure $\Rightarrow$ Components are $\{1, 2, 5, 7\}$, $\{3, 6\}$ and $\{4\}$
  $\Rightarrow$ Subgraph $\{3, 4, 6\}$ not connected, $\{1, 2, 5\}$ not maximal

Connected components

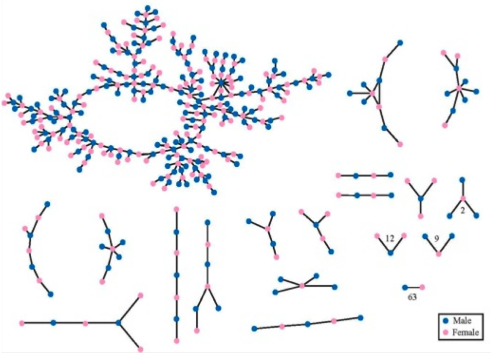
- **Def:** A component is a maximally connected subgraph
  - $\Rightarrow$ Maximal means adding a vertex will ruin connectivity



- In figure $\Rightarrow$ Components are $\{1, 2, 5, 7\}$, $\{3, 6\}$ and $\{4\}$
  $\Rightarrow$ Subgraph $\{3, 4, 6\}$ not connected, $\{1, 2, 5\}$ not maximal
- Disconnected graphs have 2 or more components
  - $\Rightarrow$ Largest component often called giant component

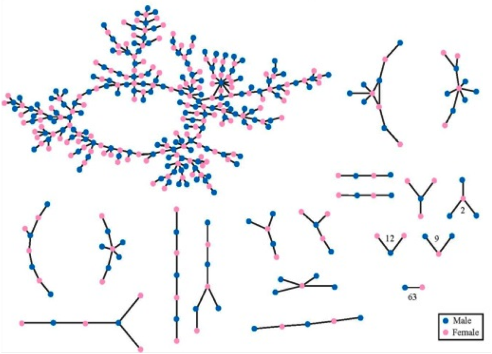
Giant connected components

- Large real-world networks typically exhibit one giant component
- Ex: romantic relationships in a US high school [Bearman et al'04]



Giant connected components

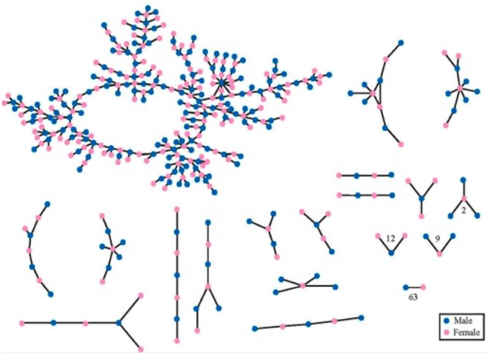
- Large real-world networks typically exhibit one giant component
- Ex: romantic relationships in a US high school [Bearman et al'04]



- Q: Why do we expect to find a single giant component?

Giant connected components

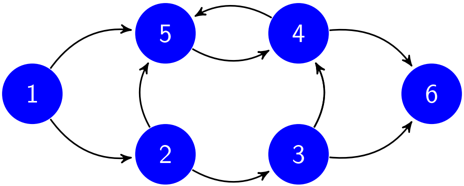
- Large real-world networks typically exhibit one giant component
- Ex: romantic relationships in a US high school [Bearman et al'04]



- Q: Why do we expect to find a single giant component?
- A: It only takes one edge to merge two giant components

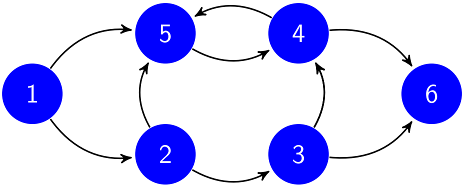
Connectivity of directed graphs

- Connectivity is more subtle with directed graphs. Two notions
- Digraph is strongly connected if for every pair $u, v \in V$, u is reachable from v (via a directed path) and vice versa
- Digraph is weakly connected if connected after disregarding edge directions, i.e., the underlying undirected graph is connected



Connectivity of directed graphs

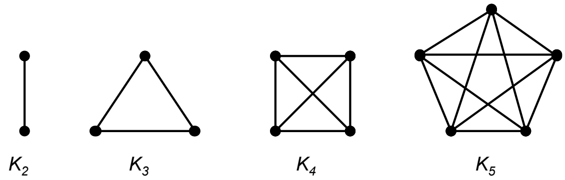
- Connectivity is more subtle with directed graphs. Two notions
- Digraph is strongly connected if for every pair $u, v \in V$, u is reachable from v (via a directed path) and vice versa
- Digraph is weakly connected if connected after disregarding edge directions, i.e., the underlying undirected graph is connected



- Above graph is weakly connected but not strongly connected

  $\Rightarrow$ Strong connectivity implies weak connectivity

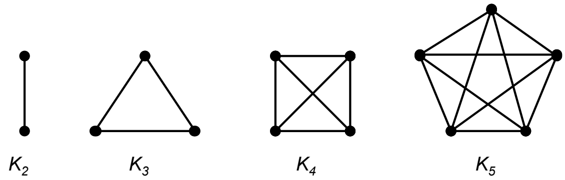
- A complete graph K_n of order n has all possible edges



- Q: How many edges does K_n have?

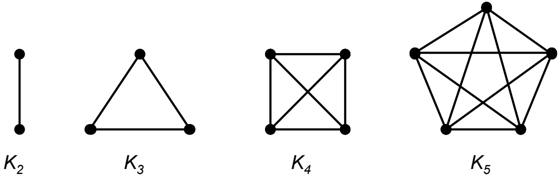
Complete graphs and cliques

- A complete graph K_n of order n has all possible edges



- Q: How many edges does K_n have?
- A: Number of edges in K_n = Number of vertex pairs = $\binom{n}{2} = \frac{n(n-1)}{2}$

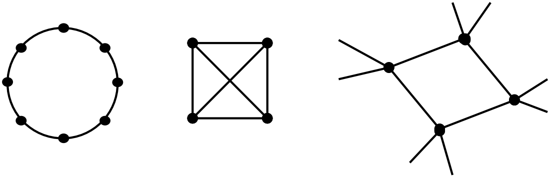
- A complete graph K_n of order n has all possible edges



- Q: How many edges does K_n have?
- A: Number of edges in K_n = Number of vertex pairs $= \binom{n}{2} = \frac{n(n-1)}{2}$
- Of interest in network analysis are cliques, i.e., complete subgraphs

  $\Rightarrow$ Extreme notions of cohesive subgroups, communities

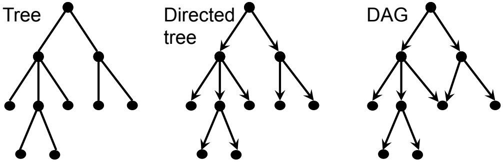
- A *d*-regular graph has vertices with equal degree d



- Naturally, the complete graph K_n is $(n-1)$-regular
    - $\Rightarrow$ Cycles are 2-regular (sub) graphs
- Regular graphs arise frequently in e.g.,
    - Physics and chemistry in the study of crystal structures
    - Geo-spatial settings as pixel adjacency models in image processing

Trees and directed acyclic graphs

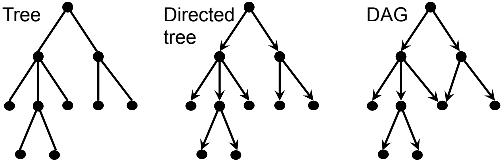
- A tree is a connected acyclic graph
  - $\Rightarrow$ A collection of trees is denominated a forest
- Ex: river network, information cascades in Twitter, citation network



- A directed tree is a digraph whose underlying undirected graph is a tree
  - $\Rightarrow$ Rooted if paths from one vertex to all others
- Vertex terminology: parent, children, ancestor, descendant, leaf

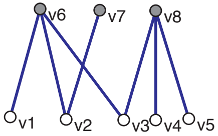
Trees and directed acyclic graphs

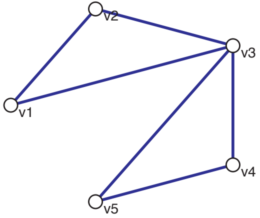
- A tree is a connected acyclic graph
  - $\Rightarrow$ A collection of trees is denominated a forest
- Ex: river network, information cascades in Twitter, citation network



- A directed tree is a digraph whose underlying undirected graph is a tree
  - $\Rightarrow$ Rooted if paths from one vertex to all others
- Vertex terminology: parent, children, ancestor, descendant, leaf
- Underlying graph of a directed acyclic graph (DAG) need not be a tree

- A graph $G(V, E)$ is called bipartite when
  - $\Rightarrow$ V can be partitioned in two disjoint sets, say V_1 and V_2; and
  - $\Rightarrow$ Each edge in E has one endpoint in V_1, the other in V_2

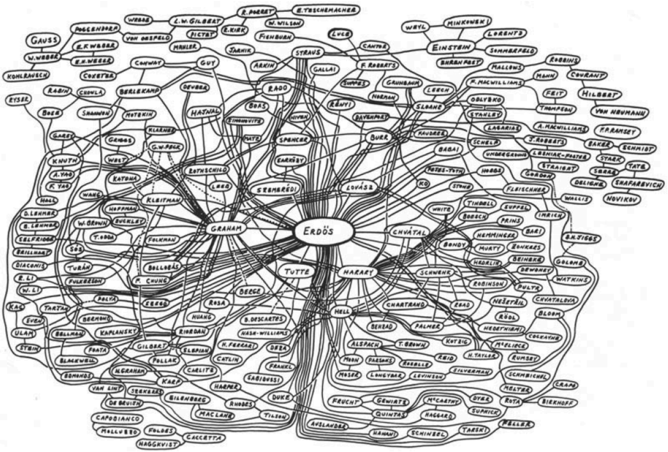




- Useful to represent e.g., membership or affiliation networks
  - $\Rightarrow$ Nodes in V_1 could be people, nodes in V_2 clubs
  - $\Rightarrow$ Associated graph $G(V_1, E_1)$ joins members of same club

Network of the week

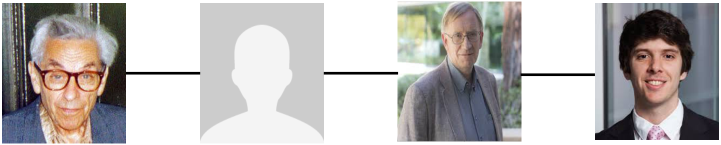
- The mathematics collaboration network centered at Paul Erdős



- Most mathematicians have an Erdős number of at most 4 or 5
  - ⇒ Drawing created by R. Graham in 1979

What is your Erdős number?

- There are resources online that can help you calculate it
- My Erdős number is 3, where a shortest path is given by



Paul Erdös — Vance Faber — Gunnar Carlsson — me

- What about the Erdős-Bacon number?
  - $\Rightarrow$ Richard Feynman 6 $(3+3)$
  - $\Rightarrow$ Carl Sagan 6 $(4+2)$
  - $\Rightarrow$ Natalie Portman 7 $(5+2)$
  - $\Rightarrow$ Santiago Segarra ∞ $(3+\infty)$ ... so far

- Algebraic graph theory deals with matrix representations of graphs

  $\Rightarrow$ Leverage algebra to 'visualize' graphs as if being plotted

Adjacency matrix

- Algebraic graph theory deals with matrix representations of graphs
  $\Rightarrow$ Leverage algebra to 'visualize' graphs as if being plotted
- Q: How can we capture the connectivity of $G(V, E)$ in a matrix?
- A: Binary, symmetric adjacency matrix $\mathbf{A} \in \{0, 1\}^{|V| \times |V|}$, with entries

$$A_{ij} = \begin{cases} 1, & \text{if } (i, j) \in E \\ 0, & \text{otherwise} \end{cases} .$$

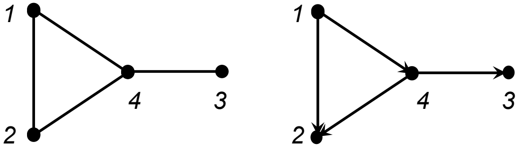
  $\Rightarrow$ Note that vertices are indexed with integers $1, \ldots, |V|$

- Algebraic graph theory deals with matrix representations of graphs
  - $\Rightarrow$ Leverage algebra to 'visualize' graphs as if being plotted
- Q: How can we capture the connectivity of $G(V, E)$ in a matrix?
- A: Binary, symmetric adjacency matrix $\mathbf{A} \in \{0, 1\}^{|V| \times |V|}$, with entries

$$A_{ij} = \begin{cases} 1, & \text{if } (i, j) \in E \\ 0, & \text{otherwise} \end{cases}.$$

  - $\Rightarrow$ Note that vertices are indexed with integers $1, \ldots, |V|$
- In words, $\mathbf{A}$ is one for those entries whose row-column indices denote vertices in V joined by an edge in E, and is zero otherwise

- Examples for undirected graphs and digraphs
  - $\Rightarrow$ There is an implicit labeling function



$$\mathbf{A}_u = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}, \quad \mathbf{A}_d = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

- If the graph is weighted, store the (i, j) weight instead of 1

- Adjacency matrix useful to store graph structure.
  - $\Rightarrow$ Also, operations on $\mathbf{A}$ yield useful information about G

Adjacency matrix properties

- Adjacency matrix useful to store graph structure.
  - $\Rightarrow$ Also, operations on $\mathbf{A}$ yield useful information about G
- Degrees: Row-wise sums give vertex degrees, i.e., $\sum_{j=1}^{|V|} A_{ij} = d_i$
- For digraphs $\mathbf{A}$ is not symmetric and row-, colum-wise sums differ

$$\sum_{j=1}^{|V|} A_{ij} = d_i^{out}, \qquad \sum_{i=1}^{|V|} A_{ij} = d_j^{in}$$

Adjacency matrix properties

- Adjacency matrix useful to store graph structure.
  $\Rightarrow$ Also, operations on $\mathbf{A}$ yield useful information about G
- Degrees: Row-wise sums give vertex degrees, i.e., $\sum_{j=1}^{|V|} A_{ij} = d_i$
- For digraphs $\mathbf{A}$ is not symmetric and row-, colum-wise sums differ

$$\sum_{j=1}^{|V|} A_{ij} = d_i^{out}, \qquad \sum_{i=1}^{|V|} A_{ij} = d_j^{in}$$

- Paths: Let $\mathbf{A}^r$ denote the r-th power of $\mathbf{A}$, with entries A_{ij}^r
  $\Rightarrow$ Then A_{ij}^r yields the number of $i - j$ paths of length r in G

Adjacency matrix properties

- Adjacency matrix useful to store graph structure.
  - $\Rightarrow$ Also, operations on $\mathbf{A}$ yield useful information about G
- Degrees: Row-wise sums give vertex degrees, i.e., $\sum_{j=1}^{|V|} A_{ij} = d_i$
- For digraphs $\mathbf{A}$ is not symmetric and row-, colum-wise sums differ

$$\sum_{j=1}^{|V|} A_{ij} = d_i^{out}, \qquad \sum_{i=1}^{|V|} A_{ij} = d_j^{in}$$

- Paths: Let $\mathbf{A}^r$ denote the r-th power of $\mathbf{A}$, with entries A_{ij}^r
  - $\Rightarrow$ Then A_{ij}^r yields the number of $i-j$ paths of length r in G
- Corollary: $\text{tr}(\mathbf{A}^2)/2 = |E|$ and $\text{tr}(\mathbf{A}^3)/6 = \#\triangle$ in G

Adjacency matrix properties

- Adjacency matrix useful to store graph structure.
  - $\Rightarrow$ Also, operations on $\mathbf{A}$ yield useful information about G
- Degrees: Row-wise sums give vertex degrees, i.e., $\sum_{j=1}^{|V|} A_{ij} = d_i$
- For digraphs $\mathbf{A}$ is not symmetric and row-, colum-wise sums differ

$$\sum_{j=1}^{|V|} A_{ij} = d_i^{out}, \qquad \sum_{i=1}^{|V|} A_{ij} = d_j^{in}$$

- Paths: Let $\mathbf{A}^r$ denote the r-th power of $\mathbf{A}$, with entries A_{ij}^r
  - $\Rightarrow$ Then A_{ij}^r yields the number of $i-j$ paths of length r in G
- Corollary: $\text{tr}(\mathbf{A}^2)/2 = |E|$ and $\text{tr}(\mathbf{A}^3)/6 = \#\triangle$ in G
- Spectrum: G is d-regular if and only if $\mathbf{1}$ is an eigenvector of $\mathbf{A}$, i.e.,

$$\mathbf{A}\mathbf{1} = d\mathbf{1}$$

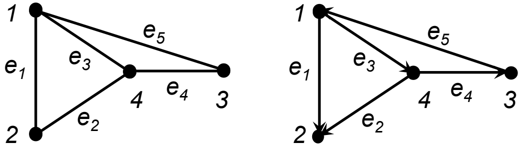
Incidence matrix

- A graph can be also represented by its $|V| \times |E|$ incidence matrix $\mathbf{B}$

  $\Rightarrow$ $\mathbf{B}$ is in general not a square matrix, unless $|V| = |E|$
- If the graph is undirected, we first assign arbitrary directions to edges
- We encode the direction of the edges, namely

$$B_{ij} = \begin{cases} 1, & \text{if edge } j \text{ is } (k, i) \\ -1, & \text{if edge } j \text{ is } (i, k) \\ 0, & \text{otherwise} \end{cases} .$$

- Example of undirected graph with added directions

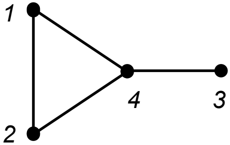


$$\mathbf{B} = \begin{pmatrix} -1 & 0 & -1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & -1 & 1 & -1 & 0 \end{pmatrix}$$

- If the graph is weighted, modify nonzero entries accordingly

- Vertex degrees often stored in the diagonal matrix $\mathbf{D}$, where $D_{ii} = d_i$

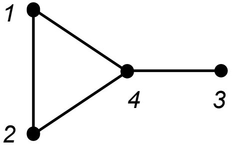
$$\mathbf{D} = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 3 \end{pmatrix}$$



Graph Laplacian

- Vertex degrees often stored in the diagonal matrix $\mathbf{D}$, where $D_{ii} = d_i$

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

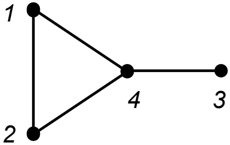


- The $|V| \times |V|$ symmetric matrix $\mathbf{L} := \mathbf{D} - \mathbf{A}$ is called graph Laplacian

$$L_{ij} = \begin{cases} d_i, & \text{if } i = j \\ -1, & \text{if } (i,j) \in E \\ 0, & \text{otherwise} \end{cases}, \quad \mathbf{L} = \begin{pmatrix} 2 & -1 & 0 & -1 \\ -1 & 2 & 0 & -1 \\ 0 & 0 & 1 & -1 \\ -1 & -1 & -1 & 3 \end{pmatrix}$$

- Vertex degrees often stored in the diagonal matrix $\mathbf{D}$, where $D_{ii} = d_i$

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



- The $|V| \times |V|$ symmetric matrix $\mathbf{L} := \mathbf{D} - \mathbf{A}$ is called graph Laplacian

$$L_{ij} = \begin{cases} d_i, & \text{if } i = j \\ -1, & \text{if } (i,j) \in E \\ 0, & \text{otherwise} \end{cases}, \quad \mathbf{L} = \begin{pmatrix} 2 & -1 & 0 & -1 \\ -1 & 2 & 0 & -1 \\ 0 & 0 & 1 & -1 \\ -1 & -1 & -1 & 3 \end{pmatrix}$$

- Variants of the Laplacian exist, with slightly different interpretations
  - $\Rightarrow$ Normalized Laplacian $\mathbf{L}_n = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$
  - $\Rightarrow$ Random-walk Laplacian $\mathbf{L}_{rw} = \mathbf{D}^{-1}\mathbf{L}$

- Smoothness: For any vector $\mathbf{x} \in \mathbb{R}^{|V|}$ of "vertex values", one has

$$\mathbf{x}^\top \mathbf{L} \mathbf{x} = \sum_{(i,j) \in E} (x_i - x_j)^2$$

which can be minimized to enforce smoothness of functions on G

Laplacian matrix properties

- Smoothness: For any vector $\mathbf{x} \in \mathbb{R}^{|V|}$ of "vertex values", one has

$$\mathbf{x}^\top \mathbf{L} \mathbf{x} = \sum_{(i,j) \in E} (x_i - x_j)^2$$

  which can be minimized to enforce smoothness of functions on G
- Incidence relation: $\mathbf{L} = \mathbf{B}\mathbf{B}^\top$ where $\mathbf{B}$ has arbitrary orientation
- Positive semi-definiteness: Follows since $\mathbf{x}^\top \mathbf{L} \mathbf{x} \geq 0$ for all $\mathbf{x} \in \mathbb{R}^{|V|}$

Laplacian matrix properties

- Smoothness: For any vector $\mathbf{x} \in \mathbb{R}^{|V|}$ of "vertex values", one has

$$\mathbf{x}^\top \mathbf{L}\mathbf{x} = \sum_{(i,j)\in E} (x_i - x_j)^2$$

  which can be minimized to enforce smoothness of functions on G
- Incidence relation: $\mathbf{L} = \mathbf{B}\mathbf{B}^\top$ where $\mathbf{B}$ has arbitrary orientation
- Positive semi-definiteness: Follows since $\mathbf{x}^\top \mathbf{L}\mathbf{x} \geq 0$ for all $\mathbf{x} \in \mathbb{R}^{|V|}$
- Rank deficiency: Since $\mathbf{L}\mathbf{1} = \mathbf{0}$, $\mathbf{L}$ is rank deficient

Laplacian matrix properties

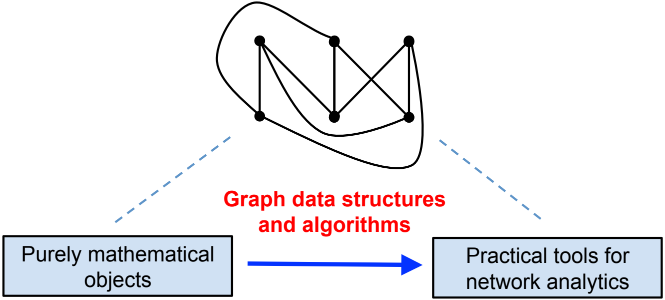
- Smoothness: For any vector $\mathbf{x} \in \mathbb{R}^{|V|}$ of "vertex values", one has

$$\mathbf{x}^\top \mathbf{L} \mathbf{x} = \sum_{(i,j) \in E} (x_i - x_j)^2$$

  which can be minimized to enforce smoothness of functions on G
- Incidence relation: $\mathbf{L} = \mathbf{B}\mathbf{B}^\top$ where $\mathbf{B}$ has arbitrary orientation
- Positive semi-definiteness: Follows since $\mathbf{x}^\top \mathbf{L} \mathbf{x} \geq 0$ for all $\mathbf{x} \in \mathbb{R}^{|V|}$
- Rank deficiency: Since $\mathbf{L}\mathbf{1} = \mathbf{0}$, $\mathbf{L}$ is rank deficient
- Spectrum and connectivity: The smallest eigenvalue λ_1 of $\mathbf{L}$ is 0
  - If the second-smallest eigenvalue $\lambda_2 \neq 0$, then G is connected
  - If $\mathbf{L}$ has n zero eigenvalues, G has n connected components

Graph data structures and algorithms

- Q: How can we store and analyze a graph G using a computer?



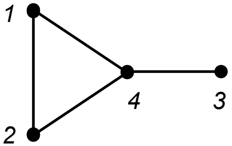
- Data structures: efficient storage and manipulation of a graph
- Algorithms: scalable computational methods for graph analytics

  $\Rightarrow$ Contributions in this area primarily due to computer science

- Q: How can we represent and store a graph G in a computer?
- A: The $|V| \times |V|$ adjacency matrix $\mathbf{A}$ is a natural choice

$$A_{ij} = \begin{cases} 1, & \text{if } (i,j) \in E \\ 0, & \text{otherwise} \end{cases}.$$

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$



- Matrices (arrays) are basic data objects in software environments
  - $\Rightarrow$ Naive memory requirement is $O(|V|^2)$
  - $\Rightarrow$ May be undesirable for large, sparse graphs

Networks are sparse graphs

- Most real-world networks are sparse, meaning

$$|E| \ll \frac{|V|(|V|-1)}{2} \text{ or equivalently } \bar{d} := \frac{1}{|V|}\sum_{v=1}^{|V|} d_v \ll |V|-1$$

- Figures from the study by Leskovec et al '09 are eloquent

Network dataset	$\lvert V \rvert$	**Avg. degree** $\bar{d}$
WWW (Stanford-Berkeley)	319,717	9.65
Social network (LinkedIn)	6,946,668	8.87
Communication (MSN IM)	242,720,596	11.1
Collaboration (DBLP)	317,080	6.62
Roads (California)	1,957,027	2.82
Proteins (S. Cerevisiae)	1,870	2.39

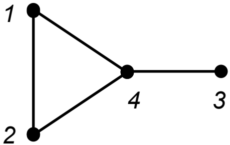
- Graph density $\rho := \frac{|E|}{|V|^2} = \frac{\bar{d}}{2|V|}$ is another useful metric

Adjacency and edge lists

- An adjacency-list representation of graph G is an array of size $|V|$

  $\Rightarrow$ The i-th array element is a list of the vertices adjacent to i

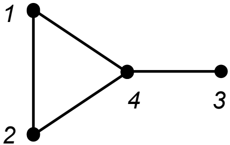
$$
\begin{aligned}
L_a[1] &= \quad \{2, 4\} \\
L_a[2] &= \quad \{1, 4\} \\
L_a[3] &= \quad \{4\} \\
L_a[4] &= \quad \{1, 2, 3\}
\end{aligned}
$$



- An adjacency-list representation of graph G is an array of size $|V|$

  $\Rightarrow$ The i-th array element is a list of the vertices adjacent to i

$$\begin{aligned} L_a[1] &= \{2,4\} \\ L_a[2] &= \{1,4\} \\ L_a[3] &= \{4\} \\ L_a[4] &= \{1,2,3\} \end{aligned}$$



- Similarly, an edge list stores the vertex pairs incident to each edge

$$\begin{aligned} L_e[1] &= \{1,2\} \\ L_e[2] &= \{1,4\} \\ L_e[3] &= \{2,4\} \\ L_e[4] &= \{3,4\} \end{aligned}$$

- In either case, the memory requirement is $O(|E|)$

Graph algorithms and complexity

- Numerous interesting questions may be asked about a given graph
- For few simple ones, lookup in data structures suffices

  Q1: Are vertices u and v linked by an edge?

  Q2: What is the degree of vertex u?

Graph algorithms and complexity

- Numerous interesting questions may be asked about a given graph
- For few simple ones, lookup in data structures suffices
  - Q1: Are vertices u and v linked by an edge?
  - Q2: What is the degree of vertex u?
- Some others require more work. Still can tackle them efficiently
  - Q1: What is the shortest path between vertices u and v?
  - Q2: How many connected components does the graph have?
  - Q3: Is a given digraph acyclic?

Graph algorithms and complexity

- Numerous interesting questions may be asked about a given graph
- For few simple ones, lookup in data structures suffices

  Q1: Are vertices u and v linked by an edge?

  Q2: What is the degree of vertex u?
- Some others require more work. Still can tackle them efficiently

  Q1: What is the shortest path between vertices u and v?

  Q2: How many connected components does the graph have?

  Q3: Is a given digraph acyclic?
- Unfortunately, in some cases there is likely no efficient algorithm

  Q1: What is the maximal clique in a given graph?
- We'll keep algorithm complexity in mind. Not focus of the course

Testing for connectivity

- Goal: verify connectivity of a graph based on its adjacency list
- Idea: start from vertex s, explore the graph, mark vertices you visit

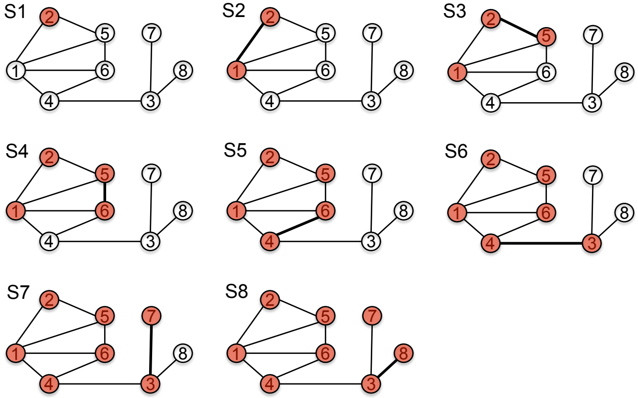
Output : List M of marked vertices in the component
Input : Graph G (e.g., adjacency list)
Input : Starting vertex s

```
L := {s}; M := {s}; % Initialize exploration and marking lists
% Repeat while there are still nodes to explore
while L ≠ ∅ do
    choose u ∈ L; % Pick arbitrary vertex to explore
    if ∃ (u, v) ∈ E such that v ∉ M then
        choose (u, v) with v of smallest index;
        L := L ∪ {v}; M := M ∪ {v}; % Mark and augment
    else
        L := L \ {u}; % Prune
    end
end
```

Graph exploration example

- Below we indicate the chosen and marked nodes. Initialize $s = 2$

L	Mark
{2}	2
{2,1}	1
{2,1,5}	5
{2,1,5,6}	6
{1,5,6}	
{1,5,6,4}	4
{5,6,4}	
{5,4}	
{5,4,3}	3
{5,3}	
{5,3,7}	7
{5,3}	
{3}	
{3,8}	8
{3}	
{}	



- Exploration takes $2|V|$ steps. Each node is added and removed once

- Choices made arbitrarily in the exploration algorithm. Variants?
- Breadth-first search (BFS): choose for u the first element of L

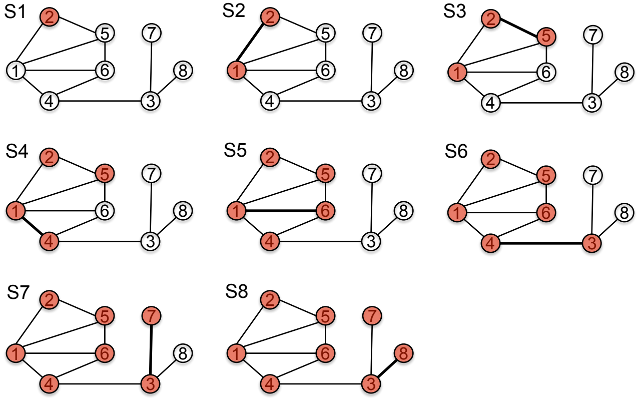
Output : List M of marked vertices in the component
Input : Graph G (e.g., adjacency list)
Input : Starting vertex s

$L := \{s\}$; $M := \{s\}$; % Initialize exploration and marking lists
% Repeat while there are still nodes to explore
while $L \neq \emptyset$ **do**
    $u :=$ first(L); % Breadth first
    if $\exists\,(u, v) \in E$ *such that* $v \notin M$ **then**
        choose (u, v) with v of smallest index;
        $L := L \cup \{v\}$; $M := M \cup \{v\}$; % Mark and augment
    else
        $L := L \setminus \{u\}$; % Prune
    end
end

BFS example

- Below we indicate the chosen and marked nodes. Initialize $s = 2$

L	Mark
{2}	2
{2,1}	1
{2,1,5}	5
{1,5}	
{1,5,4}	4
{1,5,4,6}	6
{5,4,6}	
{4,6}	
{4,6,3}	3
{6,3}	
{3}	
{3,7}	7
{3,7,8}	8
{7,8}	
{8}	
{}	



- The algorithm builds a wider tree (breadth first)

- Depth-first search (DFS): choose for u the last element of L

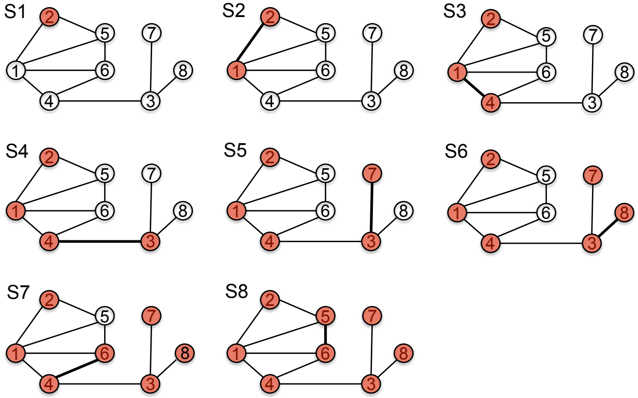
Output : List M of marked vertices in the component
Input : Graph G (e.g., adjacency list)
Input : Starting vertex s

$L := \{s\}$; $M := \{s\}$; % Initialize exploration and marking lists
% Repeat while there are still nodes to explore
while $L \neq \emptyset$ **do**
    $u :=$ last(L); % Depth first
    if $\exists\,(u, v) \in E$ *such that* $v \notin M$ **then**
        choose (u, v) with v of smallest index;
        $L := L \cup \{v\}$; $M := M \cup \{v\}$; % Mark and augment
    else
        $L := L \setminus \{u\}$; % Prune
    end
end

DFS example

- Below we indicate the chosen and marked nodes. Initialize $s = 2$

L	Mark
{2}	2
{2,1}	1
{2,1,4}	4
{2,1,4,3}	3
{2,1,4,3,7}	7
{2,1,4,3}	
{2,1,4,3,8}	8
{2,1,4,3}	
{2,1,4}	
{2,1,4,6}	6
{2,1,4,6,5}	5
{2,1,4,6}	
{2,1,4}	
{2,1}	
{2}	
{}	



- The algorithm builds longer paths (depth first)

Distances in a graph

- **Def:** The distance between vertices u and v is the length of the shortest $u - v$ path. Oftentimes referred to as geodesic distance
  - $\Rightarrow$ In the absence of a $u - v$ path, the distance is ∞
  - $\Rightarrow$ The diameter of a graph is the value of the largest distance

Distances in a graph

- **Def:** The distance between vertices u and v is the length of the shortest $u - v$ path. Oftentimes referred to as geodesic distance
  - $\Rightarrow$ In the absence of a $u - v$ path, the distance is ∞
  - $\Rightarrow$ The diameter of a graph is the value of the largest distance
- Q: What are efficient algorithms to compute distances in a graph?
- A: BFS (for unit weights) and Dijkstra's algorithm

Computing distances with BFS

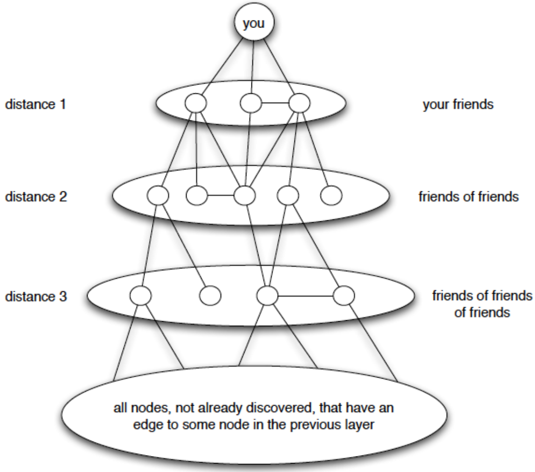
- Use BFS and keep track of path lengths during the exploration
- Increment distance by 1 every time a vertex is marked

Output : Vector d of distances from reference vertex
Input : Graph G (e.g., adjacency list)
Input : Reference vertex s

$L := \{s\}$; $M := \{s\}$; $d(s) = 0$; % Initialization
% Repeat while there are still nodes to explore
while $L \neq \emptyset$ **do**
  $u :=$ first(L); % Breadth first
  if $\exists\,(u, v) \in E$ *such that* $v \notin M$ **then**
    choose (u, v) with v of smallest index;
    $L := L \cup \{v\}$; $M := M \cup \{v\}$;% Mark and augment
    $d(v) := d(u) + 1$ % Increment distance
  else
    $L := L \setminus \{u\}$; % Prune
  end
end

Example: Distances in a social network

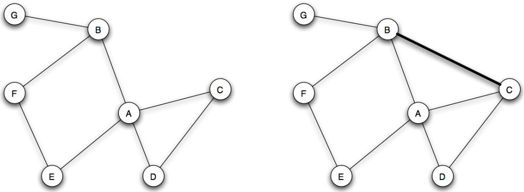
- BFS tree output for your friendship network



Triadic closure

- Networks are rarely static structures $\Rightarrow$ Think about their evolution

  $\Rightarrow$ How are edges formed? $\Rightarrow$ Universal feature $\Rightarrow$ Triadic closure
- *If two people in a social network have a friend in common, then there is an increased likelihood that they will become friends at some point in the future*



(a) *Before B-C edge forms.* (b) *After B-C edge forms.*

- Triadic closure is very natural $\Rightarrow$ Some reasons ...

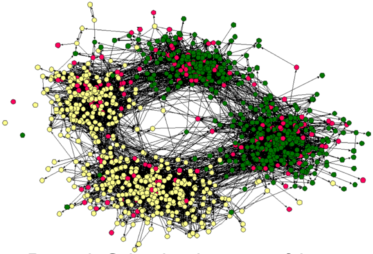
  $\Rightarrow$ Opportunity: B and C have a higher chance of meeting

  $\Rightarrow$ Trusting: B and C are predisposed to trusting each other

  $\Rightarrow$ Incentive: A might have incentive to make B and C friends

Homophily

- We tend to be similar to our friends $\Rightarrow$ Well known for long time
  - $\Rightarrow$ Age, race, interests, beliefs, opinions, affluence, ...
- Contextual (as opposed to intrinsic) effect on network formation
  - $\Rightarrow$ Contextual: Friends because we attend the same school
  - $\Rightarrow$ Intrinsic: Friends because a common friend introduces us



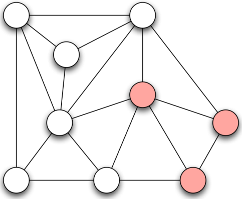
- In previous slide, B and C high chance of becoming friends
  - $\Rightarrow$ Even if they are not aware of common knowledge of A

Measuring homophily

- Is homophily present or is it an artifact of how the network is drawn?

  $\Rightarrow$ We need to formulate a precise mathematical measure

- Consider a small network of girls ($q = 3/9$) and boys ($p = 6/9$)



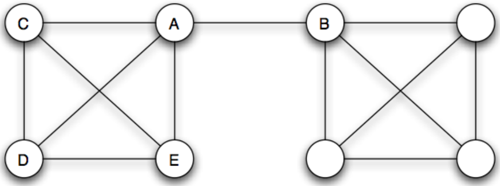
- If edges are agnostic to gender, portion of cross-gender edges is $2pq$

  $\Rightarrow$ Homophily Test: If the fraction of cross-gender edges is significantly less than 2pq, then there is evidence for homophily

  $\Rightarrow$ Cross-gender edges $5/18 < 8/18 = 2pq$ $\Rightarrow$ Mild homophily

- Mark Granovetter (1960s) interviewed people that changed jobs
- Most heard about new jobs from acquaintances rather than close friends

   $\Rightarrow$ Explanation takes into account local properties and global structure

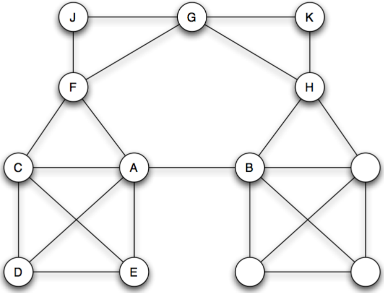


- A's friends E, C, and D form a tightly-knit group
- B reaches to a different part of the network $\Rightarrow$ New information
- Deleting (A, B) disconnects the network $\Rightarrow$ (A, B) is a bridge

   $\Rightarrow$ But bridges are rare in real-world networks

A social network closer to reality

- In real life, there are other multi-step paths joining A and B
  - $\Rightarrow$ If (A, B) is deleted, distance becomes more than 2
  - $\Rightarrow$ Local bridge
  - $\Rightarrow$ An edge is a local bridge when it is not part of a triangle

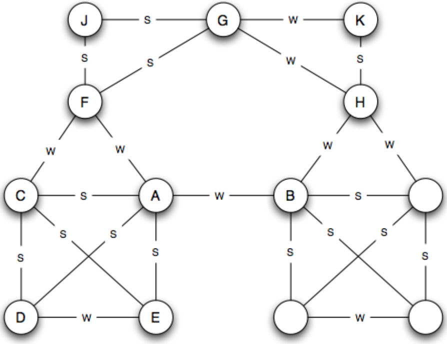


- Closely knit group of friends are eager to help
  - $\Rightarrow$ But have almost the same information as you

Strong triadic closure

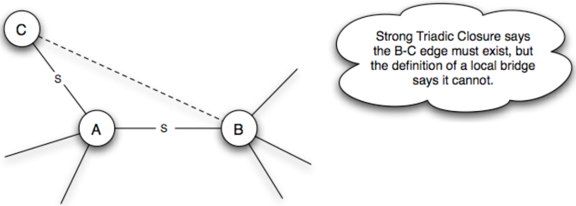
- How does overrepresentation of bridges relate to acquaintances?
- Consider two different levels of strength in the links of a social network

  $\Rightarrow$ Strong ties correspond to friends, weak ties to acquaintances



- *A* violates the Strong Triadic Closure if it has strong ties to two other nodes *B* and *C*, and there is no edge at all (strong or weak) between *B* and *C*

- Tie strength $\Rightarrow$ Local/interpersonal feature
- Bridge property $\Rightarrow$ Global/structural feature
- How do these two features relate in light of the strong triadic closure?
- *If A satisfies the strong triadic closure and is involved in at least two strong ties, then any local bridge it is involved in must be a weak tie*



- Acquaintances are natural sources of new information

  $\Rightarrow$ Strict modeling assumptions, first-order conclusions, testable