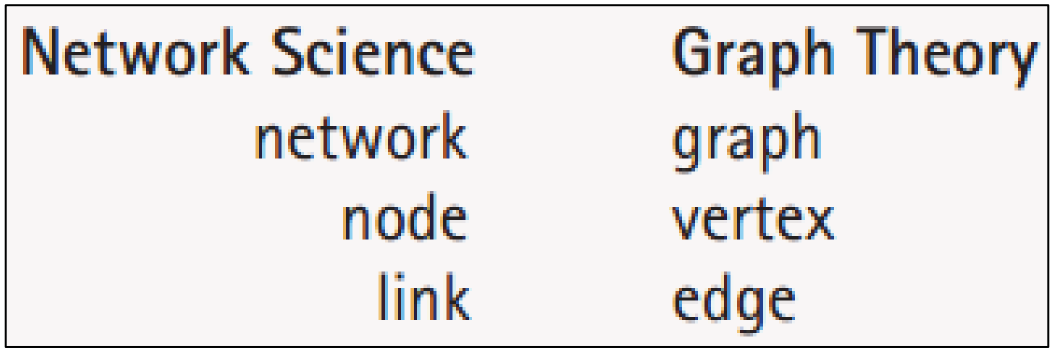
Graph Theory for Network Science

Dr. Natarajan Meghanathan
Professor
Department of Computer Science
Jackson State University, Jackson, MS
E-mail: natarajan.meghanathan@jsums.edu

Networks or Graphs

- We typically use the terms interchangeably.
- **Networks** – refers to real systems
  - WWW: network of web pages connected by URLs
  - Society: network of individuals connected by family, friendship or professional ties
  - Metabolic network: sum of all chemical reactions that take place in a cell
- **Graphs:** Mathematical representation of the networks
  - Web graph, Social graph, Metabolic graph



Network Science	Graph Theory
network	graph
node	vertex
link	edge

Real systems of quite different nature can have the same network representation

- Even though these real systems have different nature, appearance or scope, they can be represented as the same network (graph)
- Internet – connected using routers
- Actor network – network of actors who acted together in at least one movie
- Protein-Protein Interaction (PPI) network – two proteins are connected if there is experimental evidence that they can bind each other in the cell

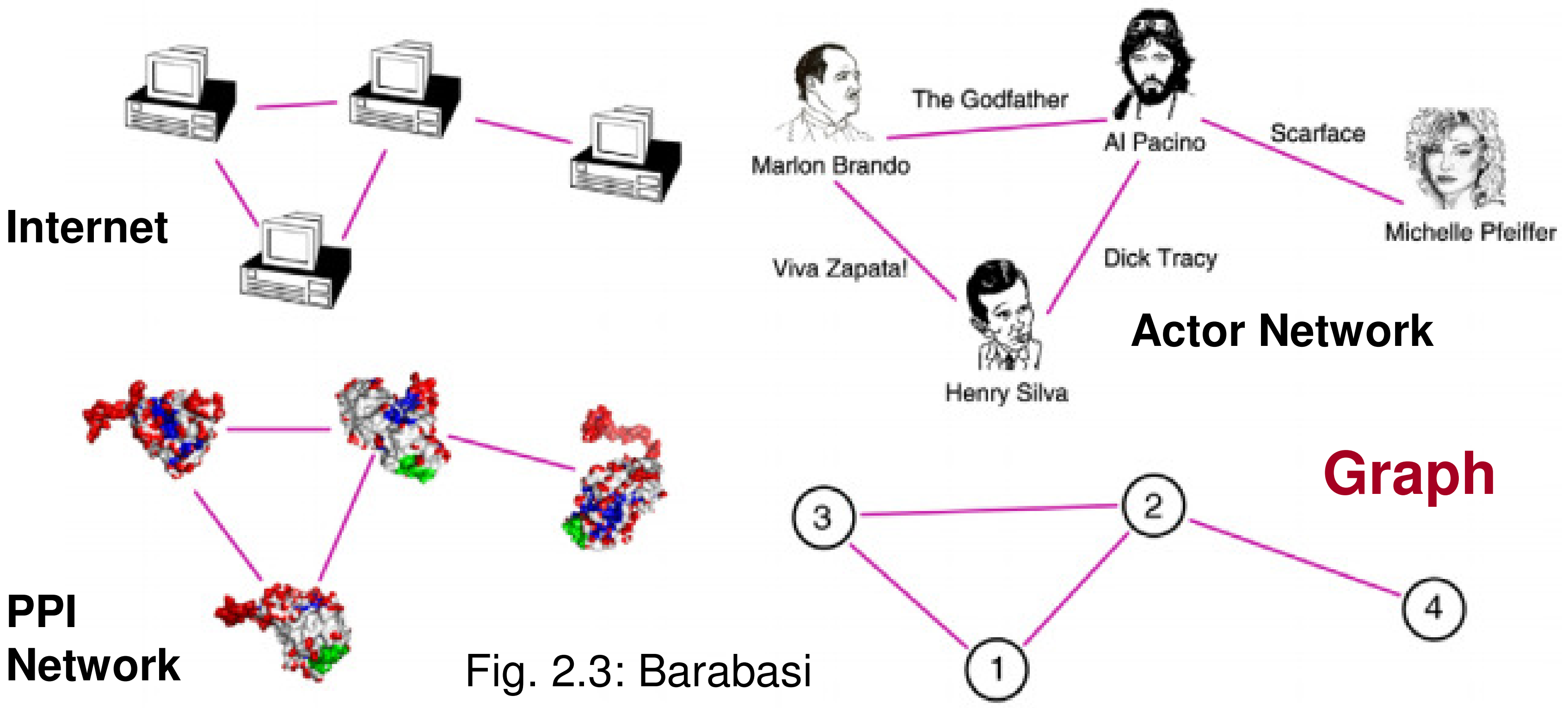


Fig. 2.3: Barabasi

Networks: Terminologies

- Node – Components in the system
- Link – Interactions between the nodes

- Directed link – nodes that interact in a specific direction (A calling B, not vice-versa; URL A is linked to URL B; A likes B)
- Undirected link – Transmission lines on the power grid (two people who are friends to each other in Facebook; electric current flowing in both directions; A and B are siblings; A and B are co-authors)
- Degree of a node – Number of links incident on it
- In-degree - # incoming links;   Out-degree - # outgoing links

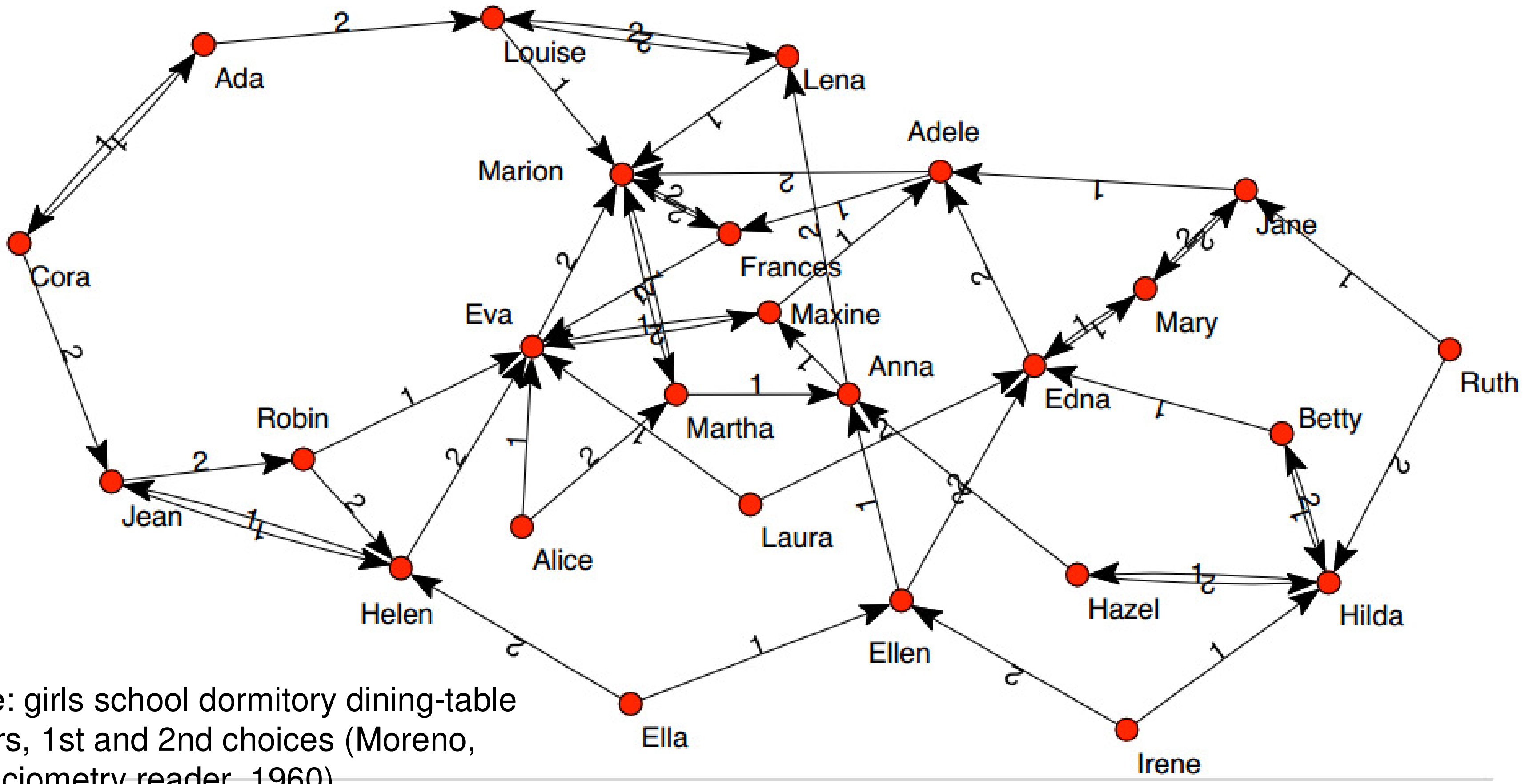
---

- Directed network – contains all directed links
- Undirected network – contains all undirected links
- Some networks can have both directed and undirected links
  - Metabolic network with certain reactions being reversible and certain reactions proceeding in only one direction
- It is important to make proper choices in the selection of links to apply the network science theory.

A regular graph is a graph in which all vertices have the same degree

Edge Attributes

- Weight (e.g., frequency of communication)
- Ranking (choice of dining parameters)
- Type (friend, relative, co-worker)

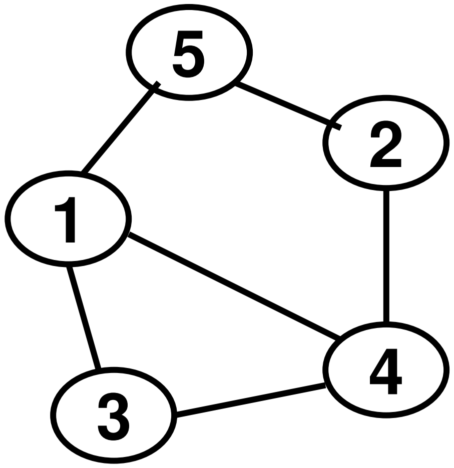


Source: girls school dormitory dining-table partners, 1st and 2nd choices (Moreno, The sociometry reader, 1960)

Storing Graph Information

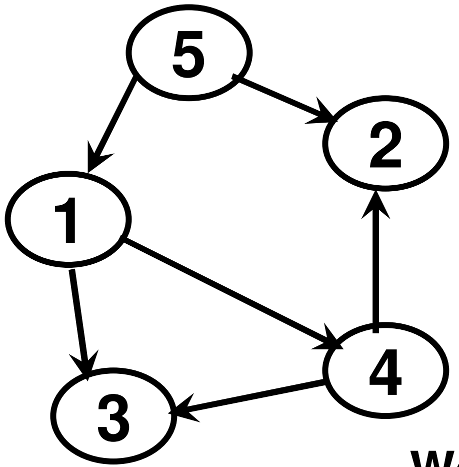
- Adjacency List

Adjacency Matrix



1	3	4	5
2	4	5	
3	1	4	
4	1	2	3
5	1	2	

$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 0 & 1 & 1 & 1 \\ 2 & 0 & 0 & 0 & 1 & 1 \\ 3 & 1 & 0 & 0 & 1 & 0 \\ 4 & 1 & 1 & 1 & 0 & 0 \\ 5 & 1 & 1 & 0 & 0 & 0 \end{array}$$



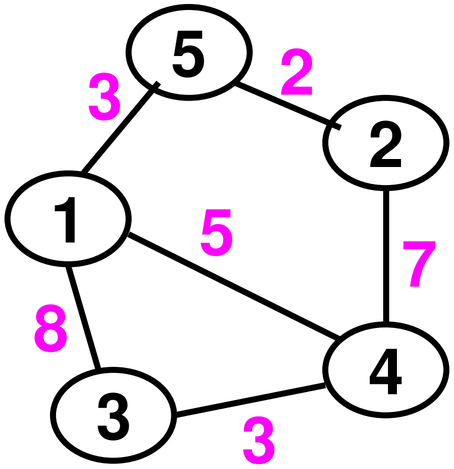
1	3	4
2		
3		
4	2	3
5	1	2

$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 0 & 1 & 1 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 1 & 1 & 0 & 0 \\ 5 & 1 & 1 & 0 & 0 & 0 \end{array}$$

We represent only the outgoing edges for directed graphs

Storing Graph Information

- Adjacency Matrix
  - Unweighted graphs: $A_{ij} = 1$ if there is a link pointing from node i to node j, and 0 otherwise
  - Weighted graphs: $A_{ij} = w_{ij}$ – weight of a link from node i to node j, and 0 otherwise



$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 0 & 8 & 5 & 3 \\ 2 & 0 & 0 & 0 & 7 & 2 \\ 3 & 8 & 0 & 0 & 3 & 0 \\ 4 & 5 & 7 & 3 & 0 & 0 \\ 5 & 3 & 2 & 0 & 0 & 0 \end{array}$$

$$A_{ij} = \begin{pmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \end{pmatrix}$$

Note that Adjacency Matrix of an Undirected graph is a symmetric matrix
i.e., $A_{ij} = A_{ji}$
for given indices i and j

Degree and # Links

On a complete graph of N nodes, the max. number of links is

$$\frac{N(N-1)}{2}$$

Average Degree = N – 1

- Undirected network
  - Let k_i denote the degree of node i, then the total number of links is:

$$L = \frac{1}{2}\sum_{i=1}^{N} k_i$$

  - The ½ factor is because we count each link twice while computing the sum of the degrees
  - The average degree of an undirected network

$$\langle k \rangle \equiv \frac{1}{N}\sum_{i=1}^{N} k_i = \frac{2L}{N}$$

For many large real-world networks, <k> ~ 1/N, implying that the networks are sparse

Degree and # Links

- Directed Network
  - Let k_i^{in} and k_i^{out} denote the incoming and outgoing degrees of node i.
  - The total number of links:

$$L = \sum_{i=1}^{N} k_i^{in} = \sum_{i=1}^{N} k_i^{out}$$

  - Average degree of a directed network is:
  -

$$\langle k^{in} \rangle = \frac{1}{N} \sum_{i=1}^{N} k_i^{in} = \langle k^{out} \rangle = \frac{1}{N} \sum_{i=1}^{N} k_i^{out} = \frac{L}{N}$$

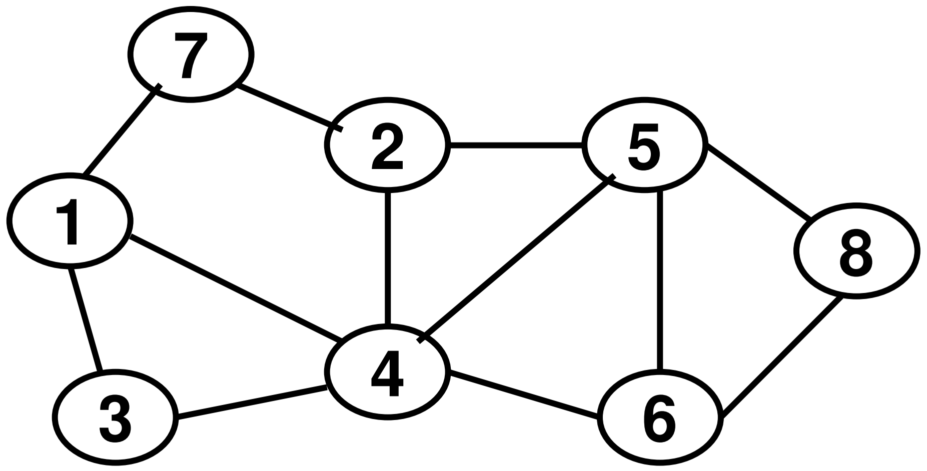
Common Network Maps: their Properties

Network Name	Nodes	Links	Directed / Undirected	# Nodes, N	# Links, L	Average Degree, <K>
E. Coli Metabolism	Metabolites	Chemical reactions	Directed	1,039	5,802	5.58
Yeast Protein Interactions	Proteins	Binding interactions	Undirected	2,018	2,930	2.90
Power Grid	Power plants, Transformers	Cables	Undirected	4,941	6,594	2.67
Science Collaboration	Scientists	Co-authorships	Undirected	23,133	186,936	16.16
Mobile Phone Calls*	Subscribers	Calls	Directed	36,595	91,826	2.51
Email	Email addresses	Emails	Directed	57,194	103,731	1.81
Internet	Routers	Internet connections	Undirected	192,244	609,066	2.67
Actor network	Actors	Co-acting	Undirected	212,250	3,054,278	28.78
WWW *	Web pages	Links	Directed	325,729	1,497,134	4.60
Citation network	Papers	Citations	Directed	449,673	4,707,958	10.47

* - Subset of the real system

Degree Distribution $\sum_{k=1}^{\infty} p_k = 1$

- Let p_k denote the probability that a randomly selected node has degree *k*.
- For a fixed number of nodes (N) in the network, $p_k = N_k / N$, where N_k is the number of degree *k* nodes.
- Average degree of a network is: $\langle k \rangle = \sum_{k=0}^{\infty} k p_k$



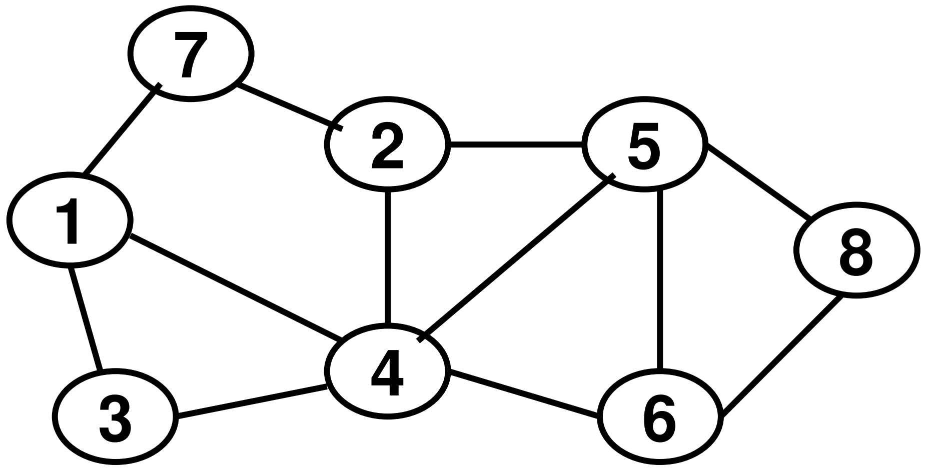
Nodes	Degree
1	3
2	3
3	2
4	5
5	4
6	3
7	2
8	2

Degree	# nodes
2	3
3	3
4	1
5	1

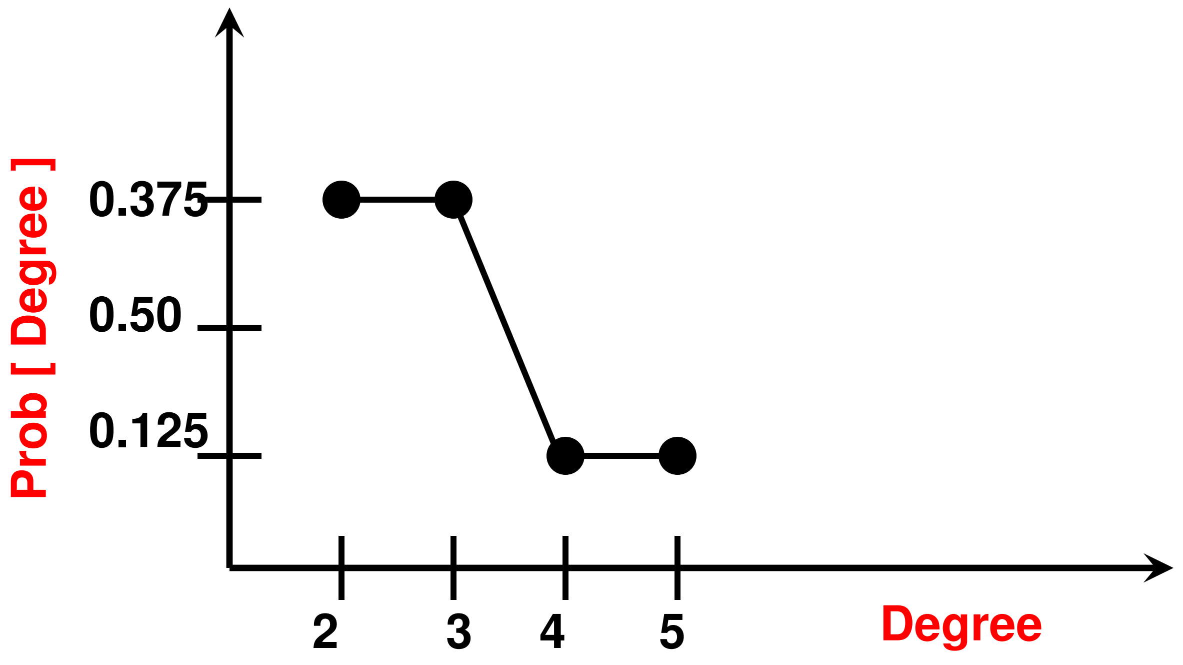
Degree	Prob[deg]
2	0.375
3	0.375
4	0.125
5	0.125

Avg. Degree = (2*0.375) + (3*0.375) + (4*0.125) + (5*0.125)
= 3.0

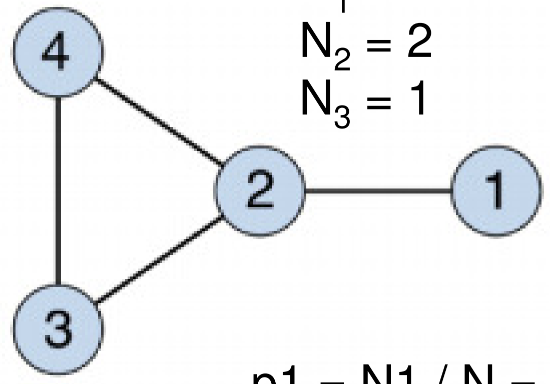
Degree Distribution



Degree	Prob[deg]
2	0.375
3	0.375
4	0.125
5	0.125

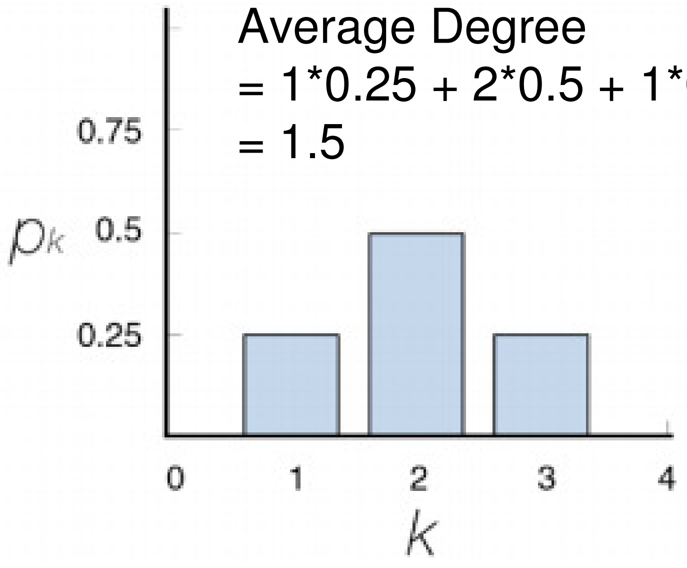


Degree Distribution Examples

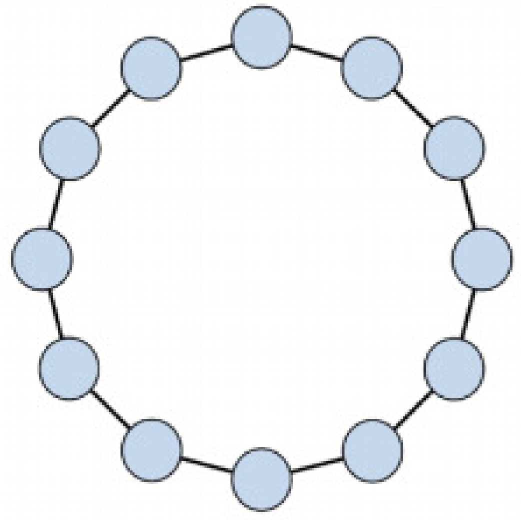


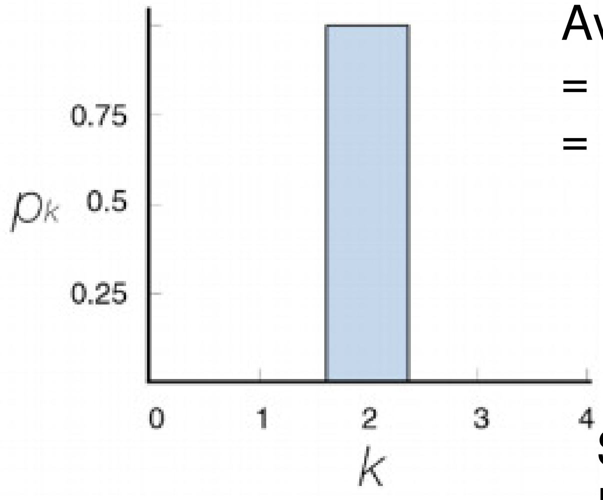
$N_1 = 1$
$N_2 = 2$
$N_3 = 1$

p1 = N1 / N = ¼ = 0.25
p2 = N2 / N = 2/4 = 0.5
p3 = N3 / N = ¼ = 0.25



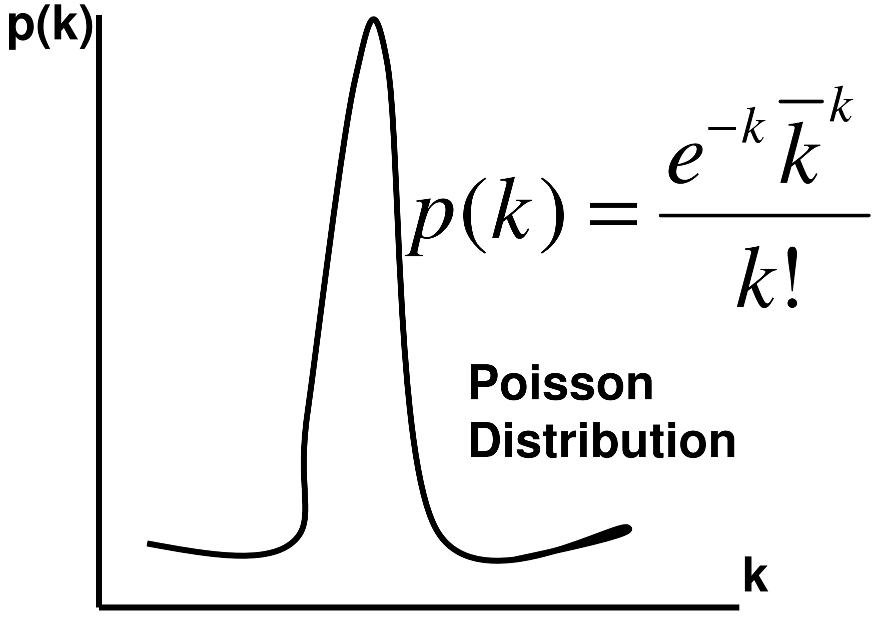
Average Degree
= 1*0.25 + 2*0.5 + 1*0.25
= 1.5



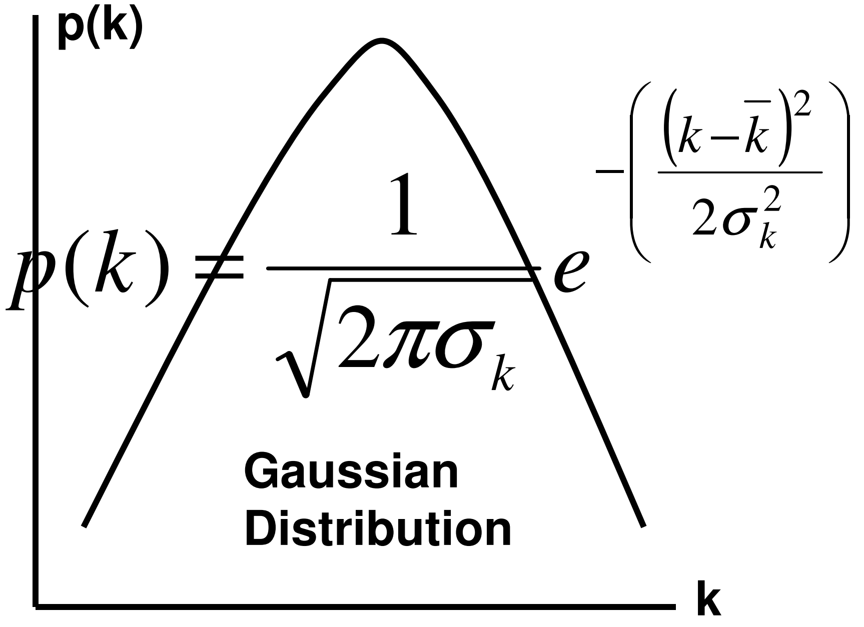


Average Degree
= 2*1.0
= 2.0

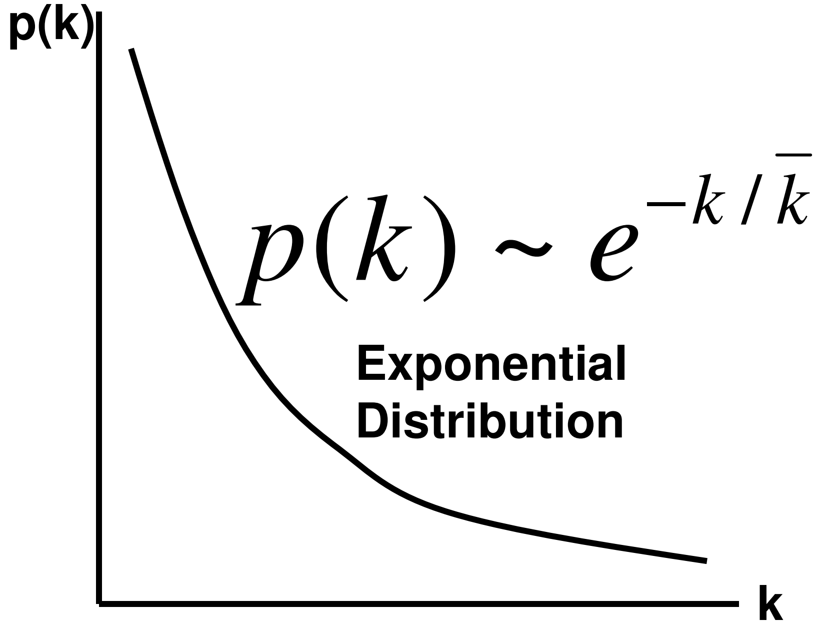
Source:
Figure 2.4a: Barabasi



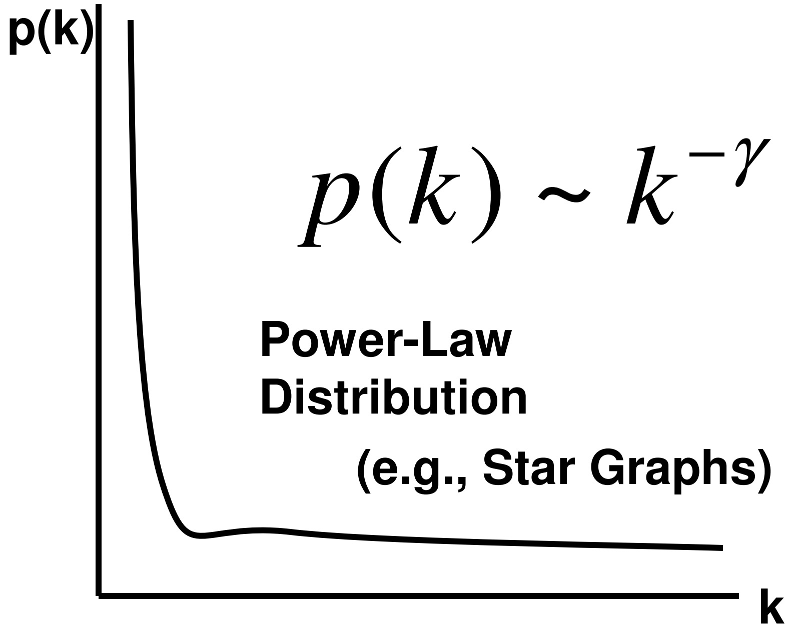
$$p(k) = \frac{e^{-k}\,\overline{k}^{k}}{k!}$$



$$p(k) = \frac{1}{\sqrt{2\pi\sigma_k}} e^{-\left(\frac{\left(k-\overline{k}\right)^2}{2\sigma_k^2}\right)}$$



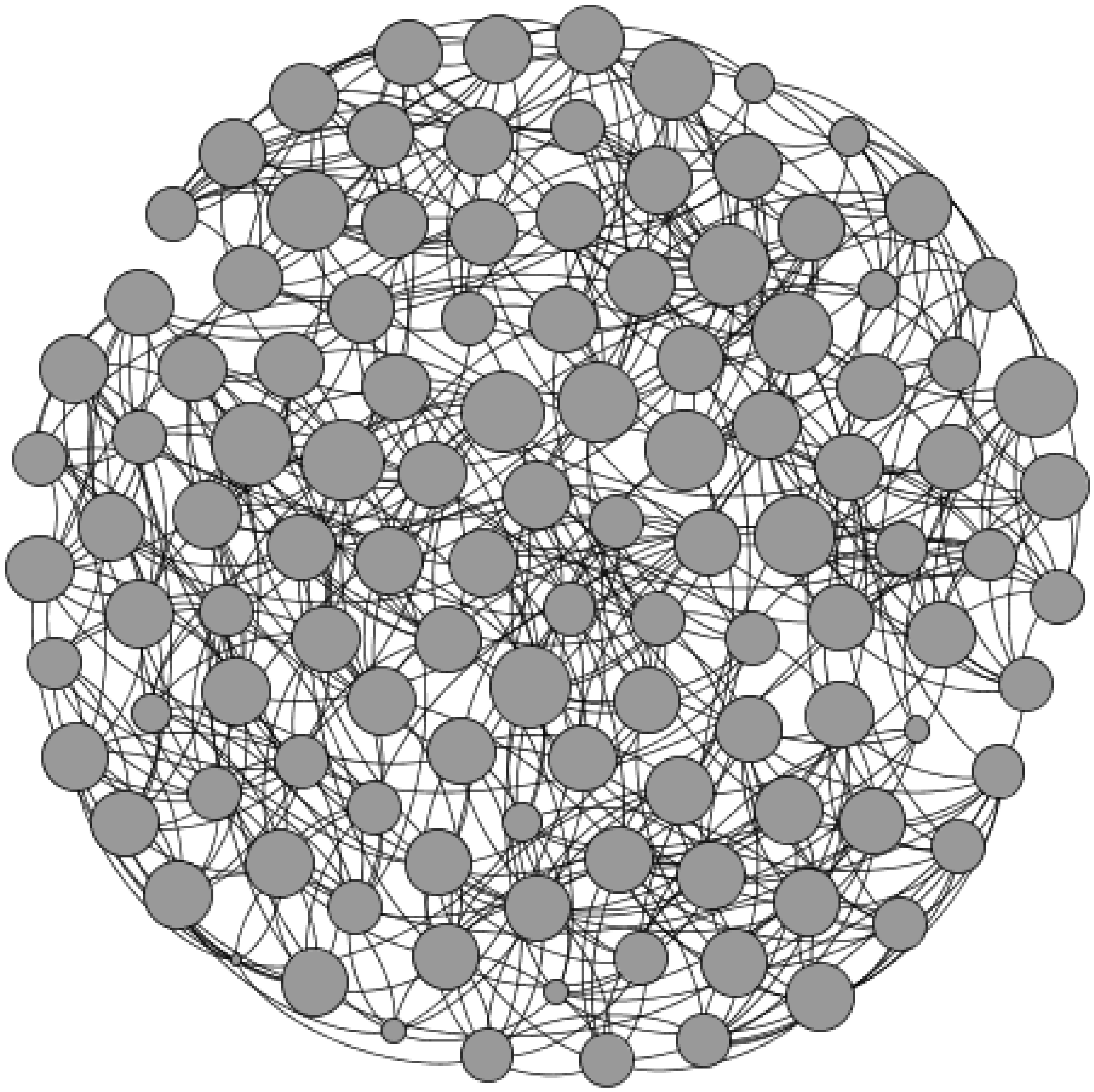
$$p(k) \sim e^{-k/\overline{k}}$$



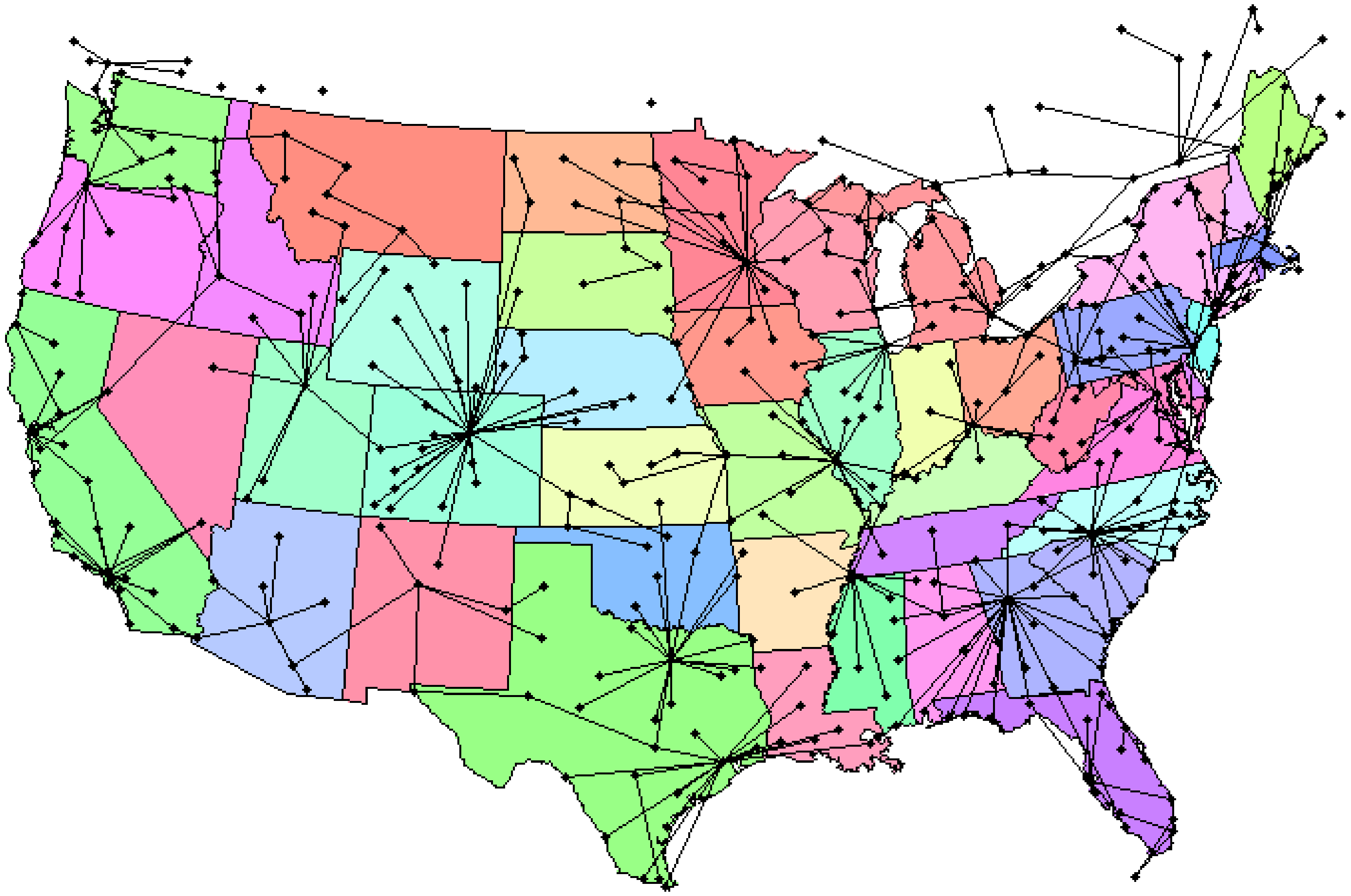
$$p(k) \sim k^{-\gamma}$$

US Football 2000 Network

This is a network of 115 football teams (nodes) that competed in the Fall 2000 season. There is an edge between two teams (nodes) if they have competed against each other during the season.

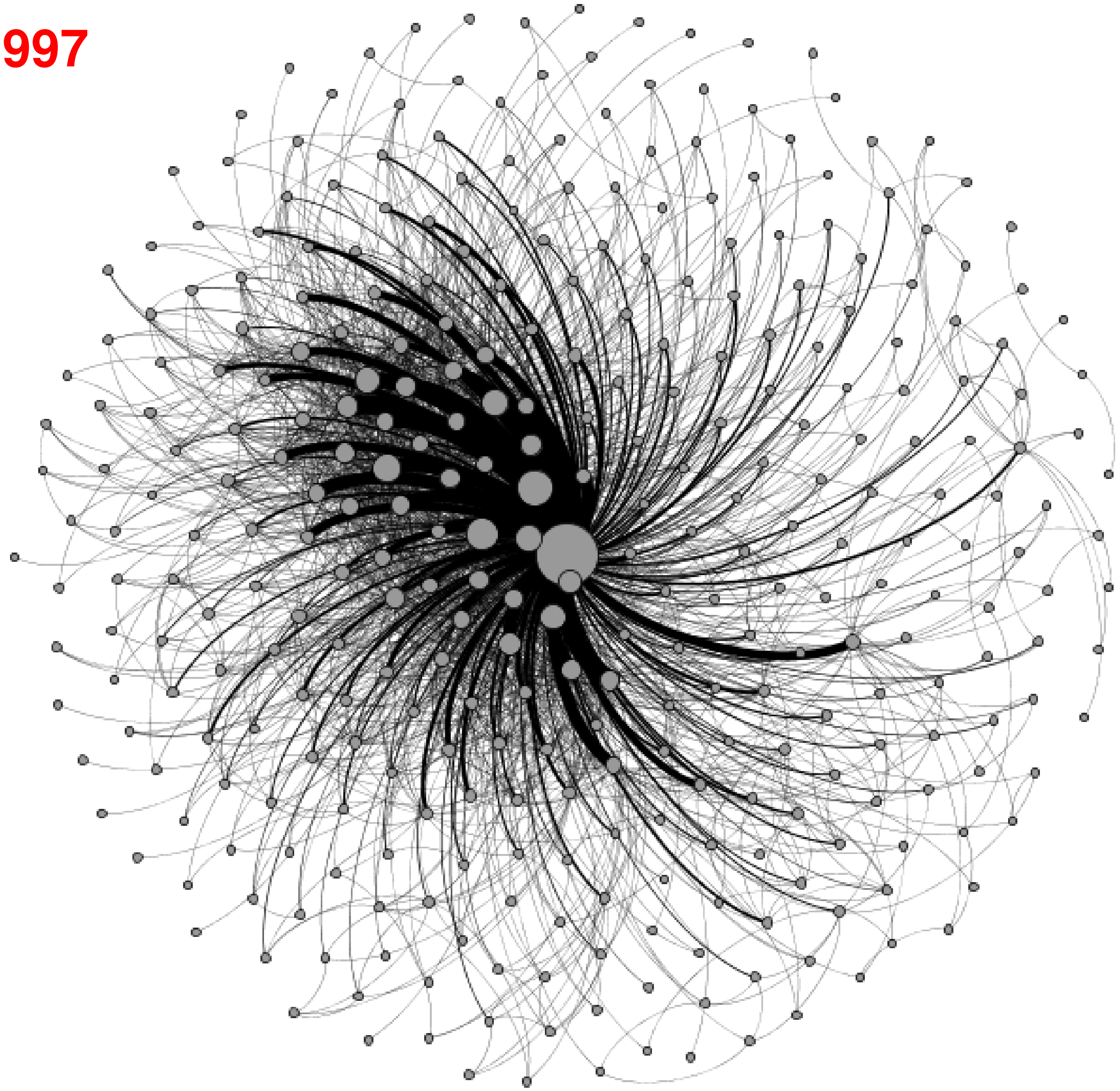


US Airports network

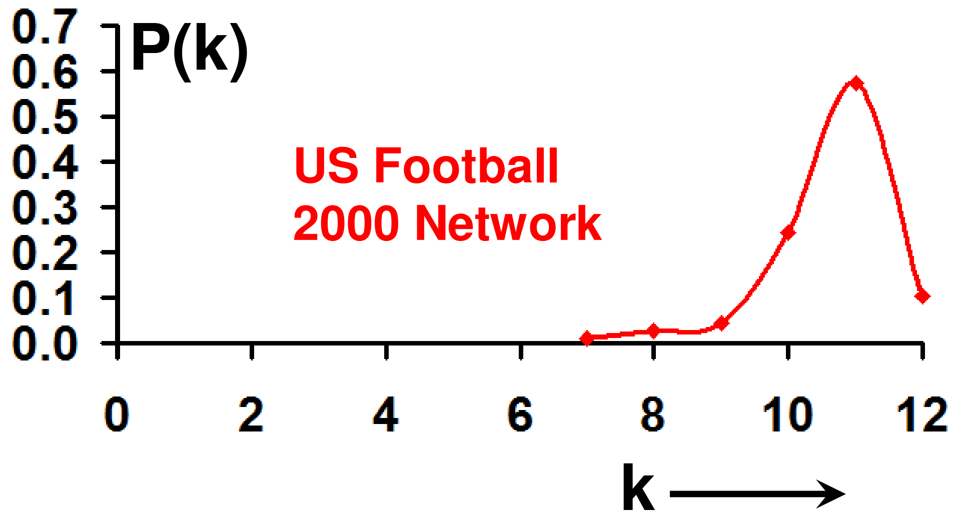


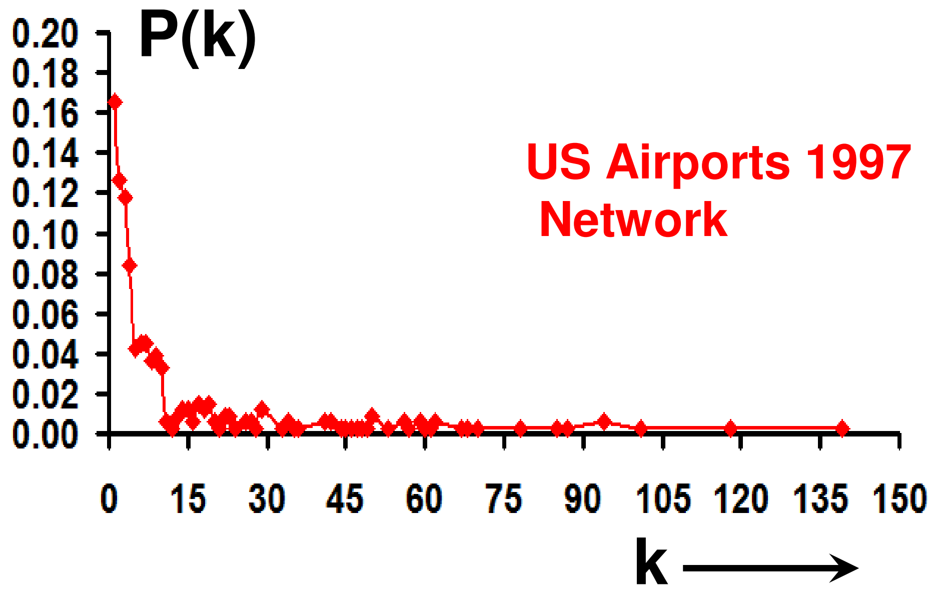
US Airports 1997 Network

This is a network of 332 airports in the US in 1997. There is an edge between two airports if there is a direct flight connection between them.



Degree Distribution of Real-World Networks



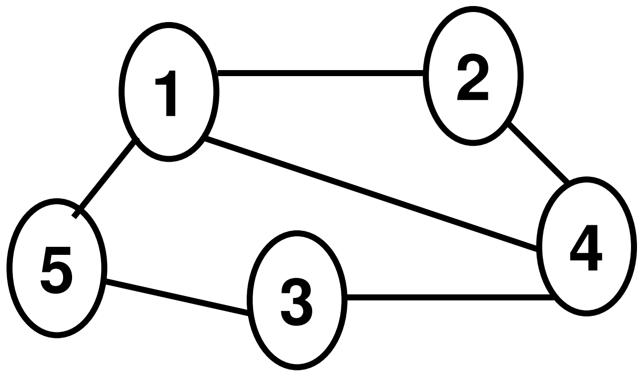


Assortativity Index

- Assortativity is a measure of the association of nodes of similar weight (usually, refer to node degrees). That is, high degree nodes tend to associate with high degree nodes and low-degree nodes with low-degree nodes.
- On the other hand, if high degree nodes associate with low degree nodes and vice-versa, it is referred to as disassortativity.
- We measure the assortativity index as the Pearson Correlation Coefficient (r) evaluated on the degrees of the end nodes of every link in the network.
  - Positive values of r indicate the network exhibits assortativity.
  - Negative values of r indicate the network exhibits diassortativity.
  - Values of r close to 0 indicates the network is more neutral.

$$r = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}}$$

Example: Assortativity Index



$$r = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}}$$

We follow the convention of considering edges in the increasing order of the left node ID, followed by increasing order of right node ID.

Edges	**Degrees of End Nodes**					
X – Y	**Degree(X)**	**Degree(Y)**		**X – Avg(X)**	**Y – Avg(Y)**	**(X-Avg(X))(Y-Avg(Y))**
1 – 2	**3**	**2**		**0.5**	**-0.5**	**-0.25**
1 – 4	**3**	**3**		**0.5**	**0.5**	**0.25**
1 – 5	**3**	**2**		**0.5**	**-0.5**	**-0.25**
2 – 4	**2**	**3**		**-0.5**	**0.5**	**-0.25**
3 – 4	**2**	**3**		**-0.5**	**0.5**	**-0.25**
3 – 5	**2**	**2**		**-0.5**	**-0.5**	**0.25**
Avg. X	**2.5**	**2.5**	**SumSq**	**1.5**	**1.5**	**Sum -0.5**

Assortativity Index = -0.5 / [sqrt(1.5) * sqrt(1.5)] = -0.333 [disassortativity]

Assortativity of Social Networks

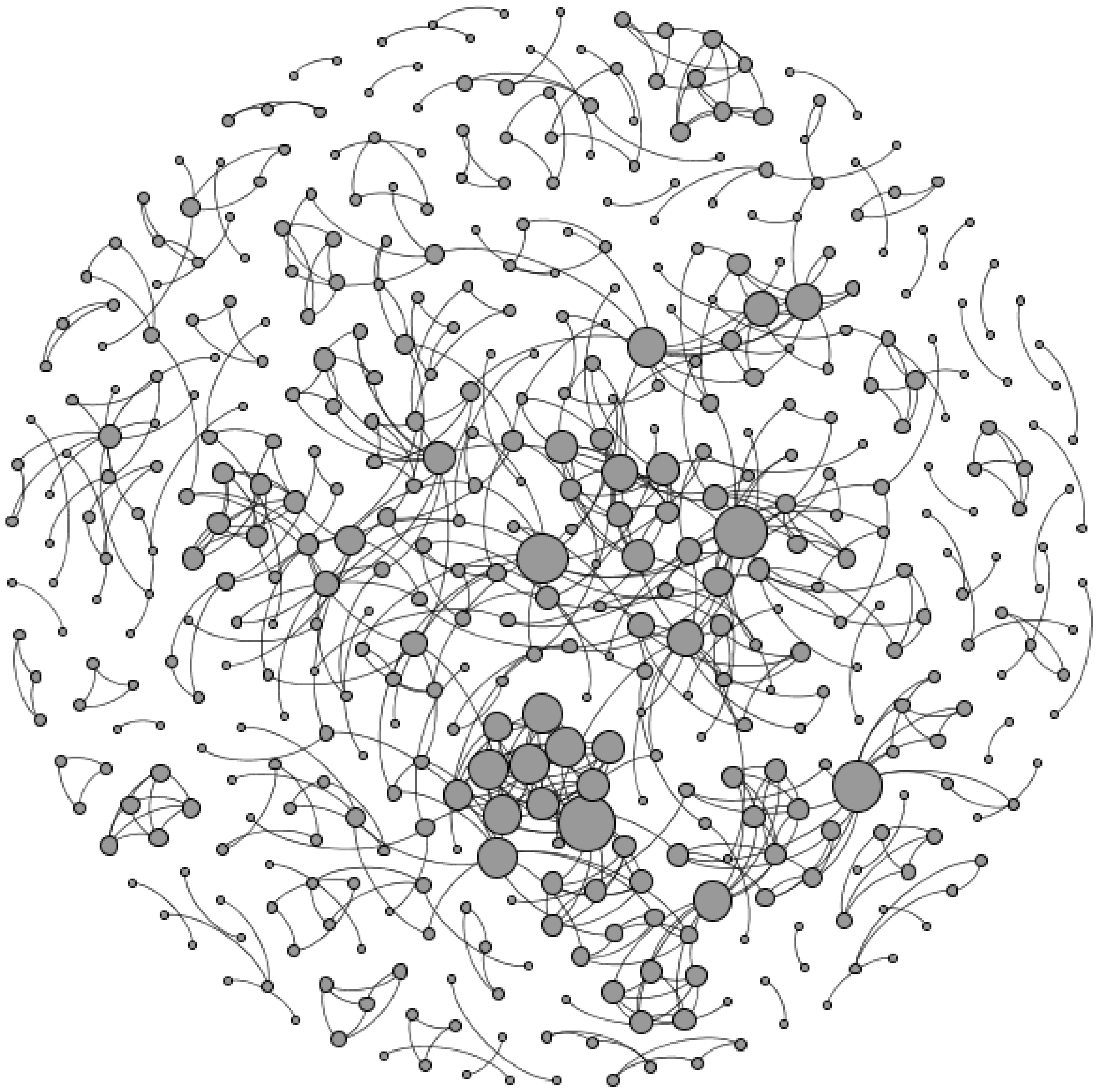
Network	# Nodes	Assortativity Index
Physics Co-authorship	52,909	0.363
Film actor Collaborations	449,913	0.208
Company Directors	7,673	0.276
E-mail Comm.		

In real world, most of the social networks are assortative and the non-social Networks are typically disassortative. However, there are some exceptions.

Network	Assortativity Index	Network	Assortativity Index
Drug Users	**-0.118**	**Roget's Thesaurus**	**0.174**
Karate Club	**-0.476**	**Protein Structure**	**0.412**
Students dating	**-0.119**	**St Marks Food Web**	**0.118**

Assortativity of Social Journal Net.

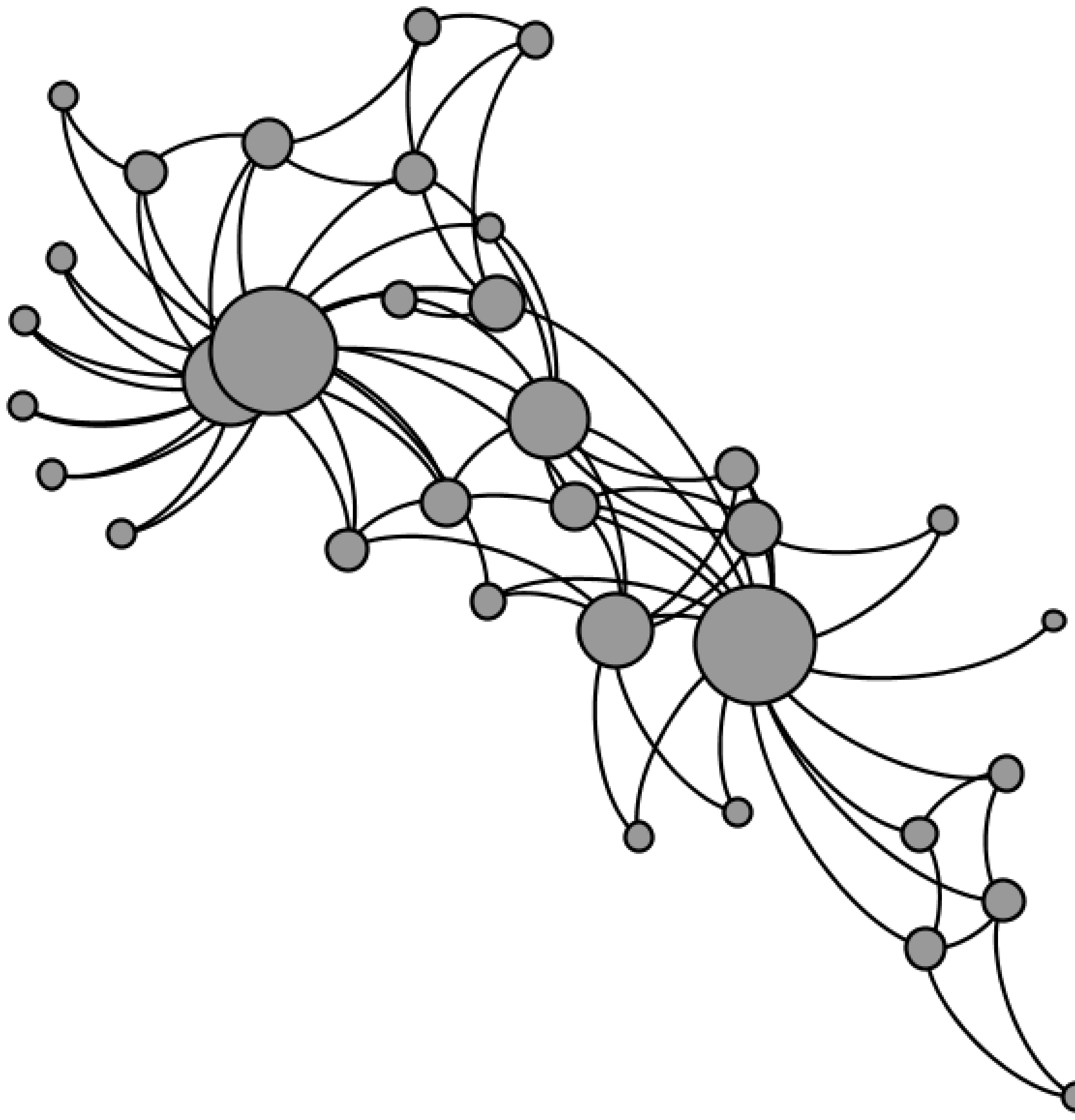
This is a network of 475 authors (vertices) involved in the production of 295 articles for the Social Networks Journal since its inception until 2008; there is an edge between two vertices if the corresponding authors co-authored at least one paper published in the journal.



Assortative Index 0.349

Assortativity of Karate Club Network

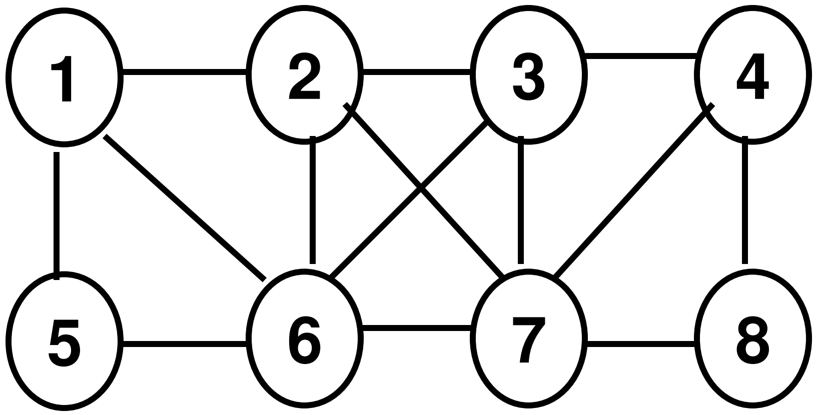
This is a network of 34 members (nodes) of a Karate Club at a US university in the 1970s; there is an edge between two nodes if the corresponding members were seen interacting with each other during the observation period.



Assortative Index - 0.476

Maximal Matching of Edges

- A "Matching" M for a graph G = (V, E) is a set of independent edges (chosen from E) such that no two edges in M have a common end vertex.
- A "Maximal Matching" is a set of independent edges such that the addition of one more edge to this set violates the property of matching.
- A "Maximum Matching" is a set of independent edges such that every vertex in the graph could be paired with another vertex without violating the property of matching.



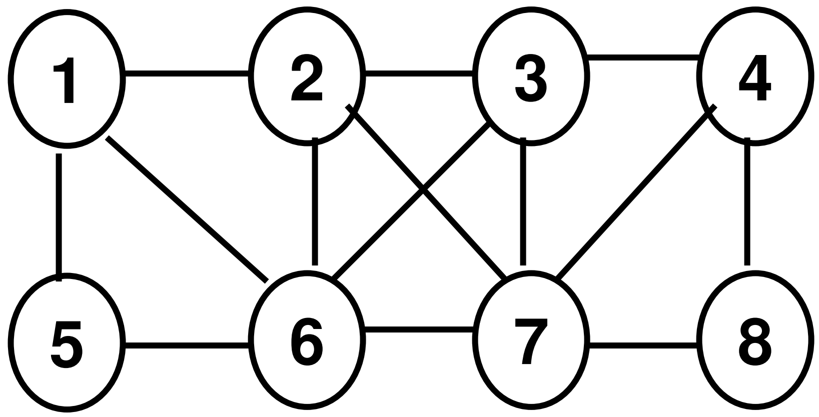
Matching (arbitrary): {1 – 2, 3 – 4}

Maximal Matching	**{1 – 5, 2 – 7, 3 – 4}**
	{1 – 6, 2 – 7, 4 – 8}
	{1 – 5, 2 – 6, 3 – 7, 4 – 8}
	{1 – 2, 3 – 4, 5 – 6, 7 – 8}
Maximum Matching	**{1 – 5, 2 – 6, 3 – 7, 4 – 8}**
	{1 – 2, 3 – 4, 5 – 6, 7 – 8}

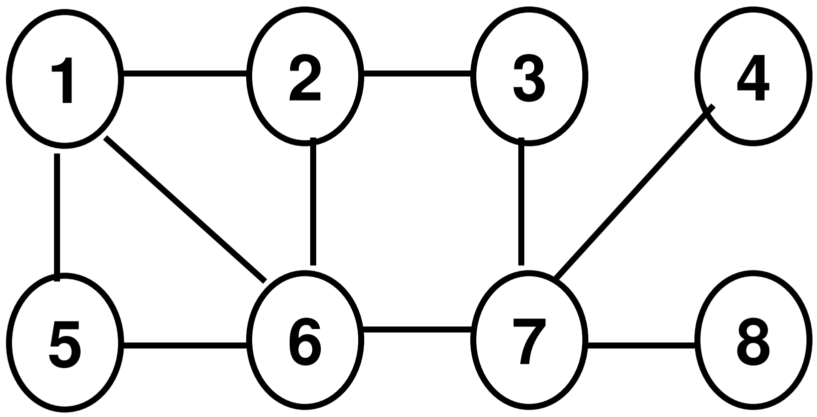
For a graph with odd number of vertices (V), the maximum number of node pairs that could be matched is (V/2) – 1.

Maximal Node Matching (MNM)

- As a maximal matching need not maximize the number of nodes matched, the objective for MNM is to maximize the number of nodes that could be matched.
- If the MNM of a graph contains all the vertices in the graph, then the MNM corresponds to a maximum matching.
- **Sample Application:** Given a network of people who can work with each other, we want to maximize the number of two-member teams.



{1 – 2, 3 – 4, 5 – 6, 7 – 8} is an MNM as well as a maximum matching

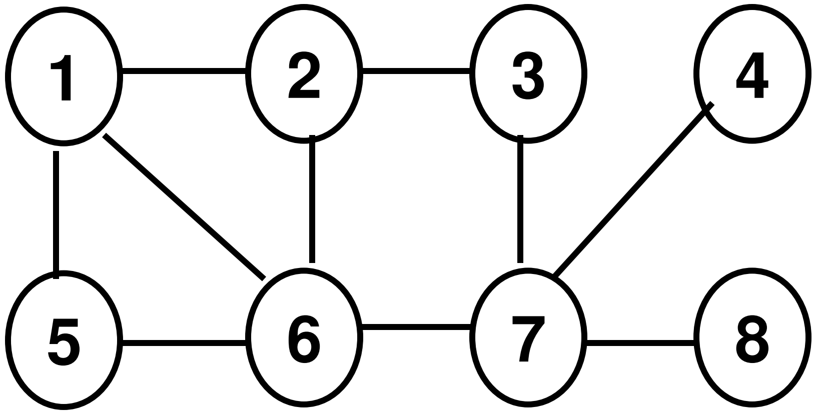


{1 – 5, 2 – 6, 3 – 7} is an MNM, but not a maximum matching

{1 – 2, 6 – 7} is a maximal matching, but is not an MNM.

MNM Algorithm

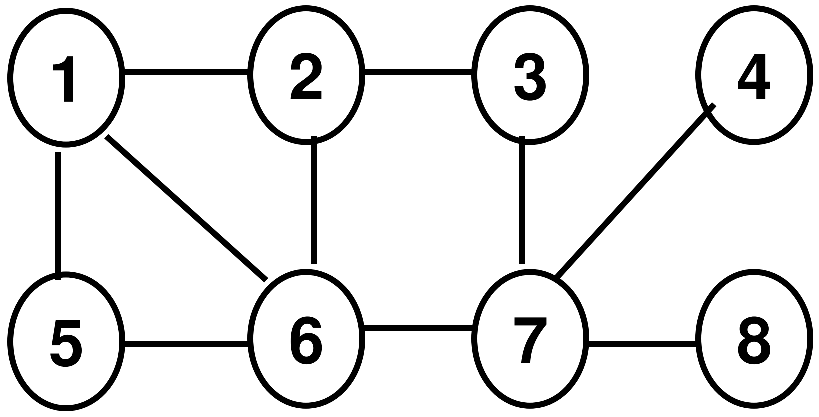
- Let M be the set of edges constituting a matching (MNM).
- An edge is said to be covered if it is either in M or is adjacent to an edge in M. **Two edges are adjacent if they have a common end vertex**
- At any time, we define the coverage weight of an edge as the number of uncovered edges adjacent to it.
- To start with, each edge in the graph is an uncovered edge.
- In each iteration, we remove the edge with the minimum coverage weight and include in the set M and also remove its adjacent edges (as they are no longer uncovered).
  - **We prefer to include an edge with a minimum coverage weight as such an edge is likely to cover a minimal number of adjacent edges and we can maximize the number of nodes matched.**
- We repeat the iterations until there exists no edge in the graph.



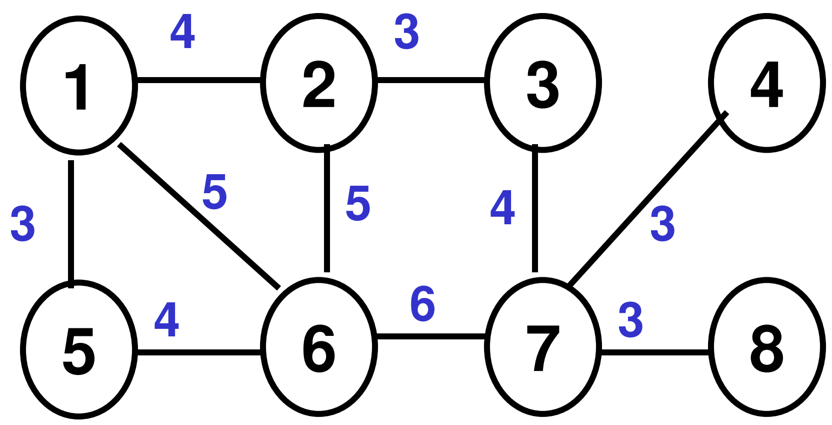
Assume all edges are initially uncovered

Edge	Coverage Weight	Edge	Coverage Weight
1 – 2	4	3 – 7	4
1 – 5	3	4 – 7	3
1 – 6	5	5 – 6	4
2 – 3	3	6 – 7	6
2 – 6	5	7 – 8	3

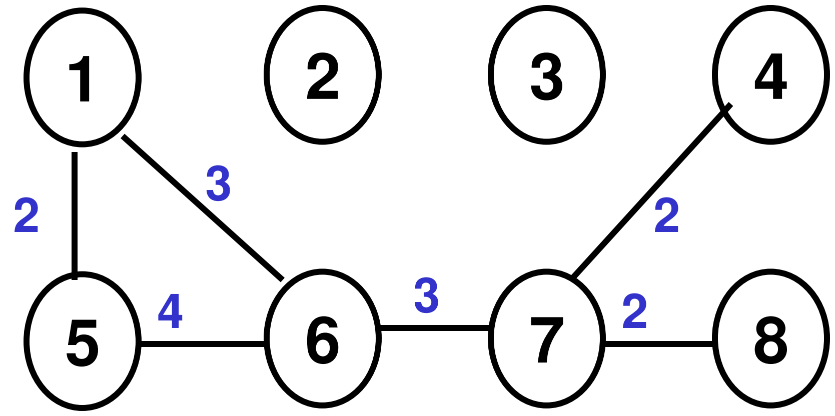
MNM Example 1



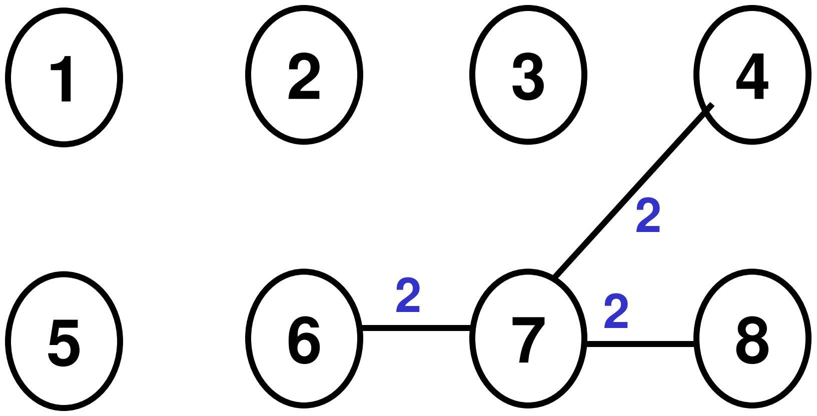
Given Graph



Initial Coverage Weights

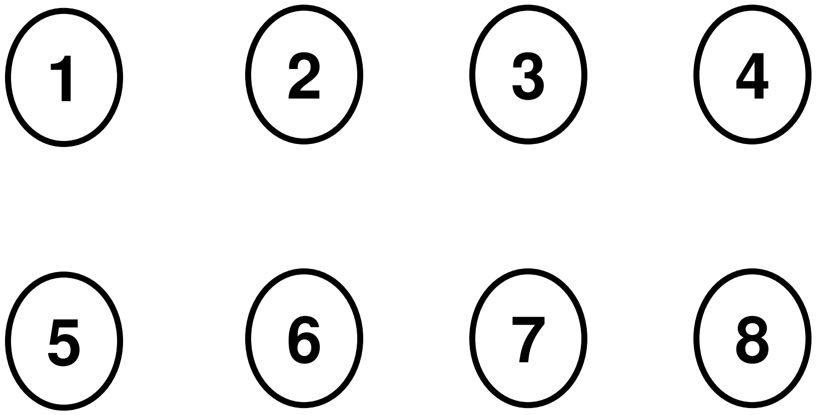


Iteration 1
(Remove Edge 2 – 3
and its adjacent edges)

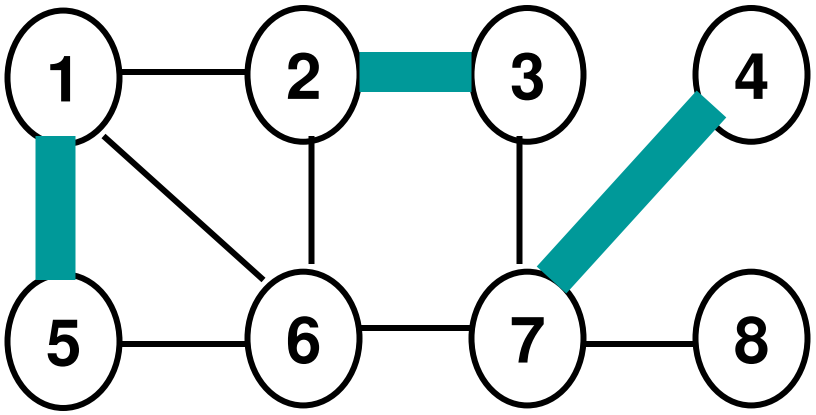


Iteration 2
(Remove Edge 1 – 5
and its adjacent edges)

MNM Example 1 (2)



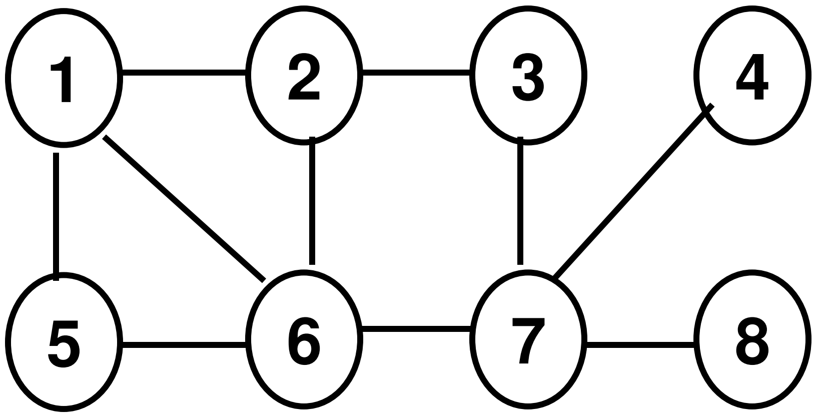
Iteration 3
(Remove Edge 4 – 7
and its adjacent edges)



Final Maximal Node Matching

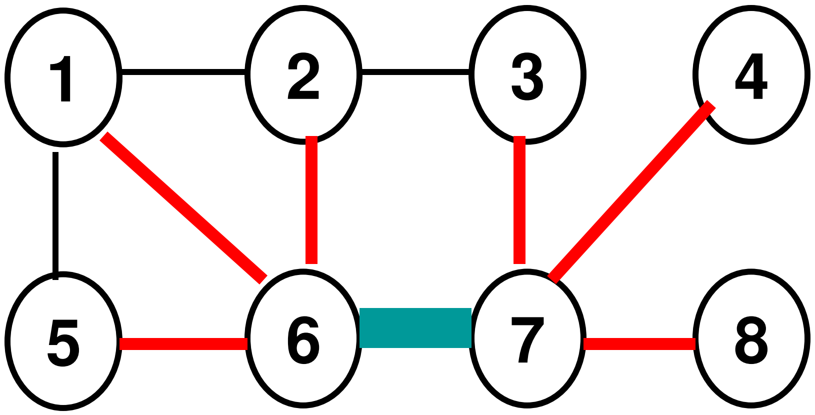
Intuition for Edge Selection Criteria

- What happens if pick an edge with the largest coverage weight to be part of MNM?



Assume all edges are initially uncovered

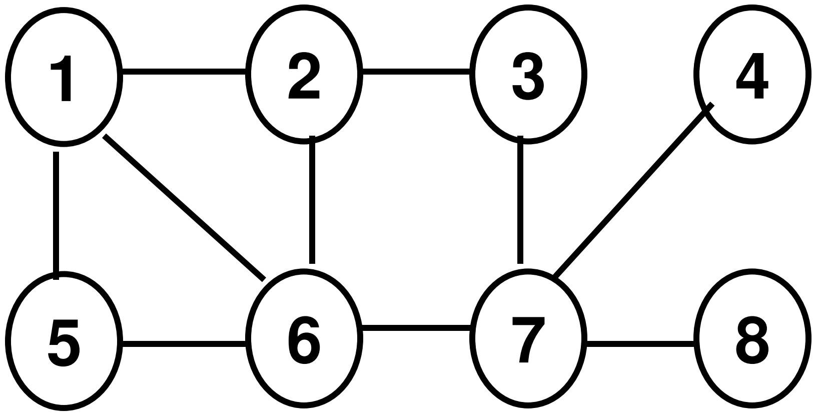
Edge	Coverage Weight	Edge	Coverage Weight
1 – 2	4	3 – 7	4
1 – 5	3	4 – 7	3
1 – 6	5	5 – 6	4
2 – 3	3	6 – 7	6
2 – 6	5	7 – 8	3



If select the edge {6 – 7} with the largest coverage weight of 6 to be part of the MNM, then we see that we are losing several edges from being considered for a matching. There are only three available edges {1 – 2; 1 – 5; 2 – 3} for a subsequent iteration.

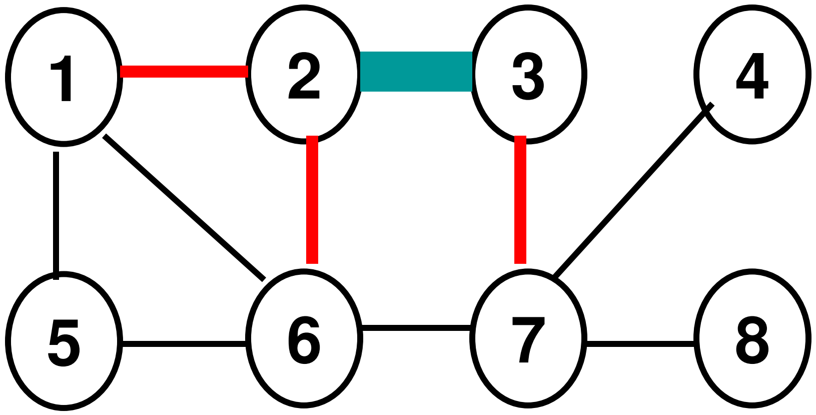
Intuition for Edge Selection Criteria (2)

- What happens if pick an edge with the smallest coverage weight to be part of MNM?



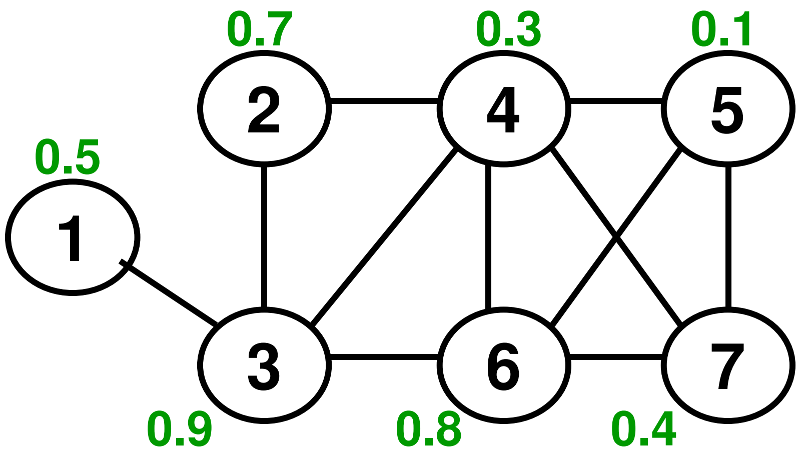
Assume all edges are initially uncovered

Edge	Coverage Weight	Edge	Coverage Weight
1 – 2	4	3 – 7	4
1 – 5	3	4 – 7	3
1 – 6	5	5 – 6	4
2 – 3	3	6 – 7	6
2 – 6	5	7 – 8	3

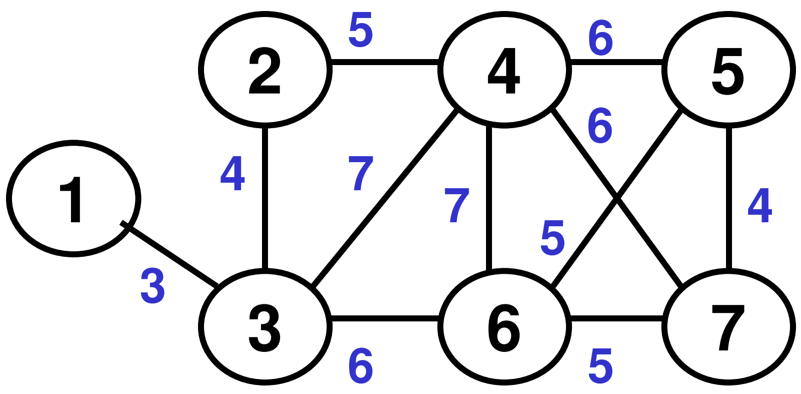


If select the edge {2 – 3} with the smallest coverage weight of 3 to be part of the MNM, then we are likely to retain several edges to be considered to be part of a matching in the subsequent iterations.

MNM Example 2

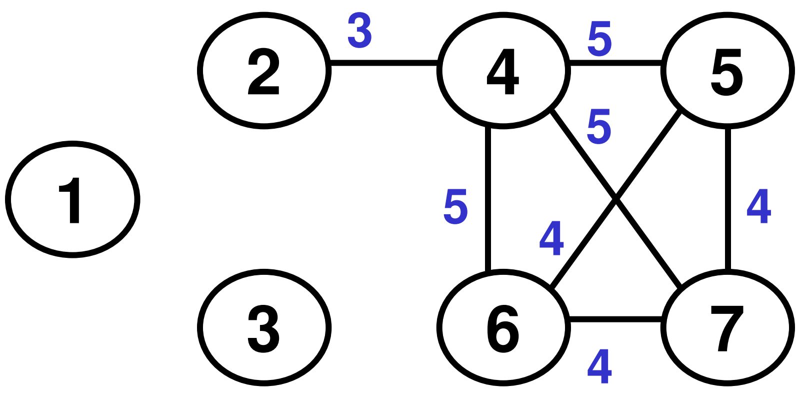


Given Graph

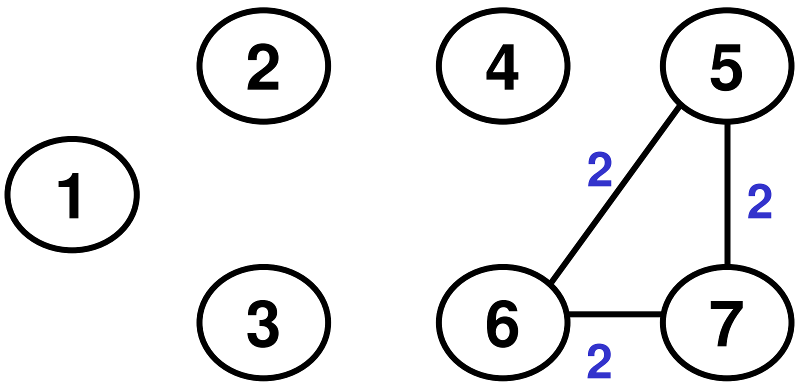


Initial Coverage Weights

Note that in the case of MNM, we do not consider the weights of the vertices (even if they have some weight) while computing the coverage weights



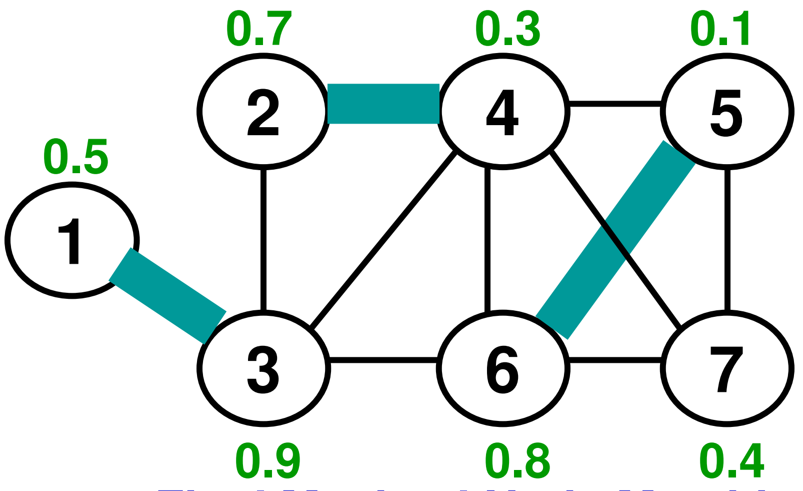
Iteration 1
(Remove Edge 1 – 3
and its adjacent edges)



Iteration 2
(Remove Edge 2 – 4
and its adjacent edges)

MNM Example 2 (2)

% of Node Matches: 86%
Assortative Index: – 0.55



Final Maximal Node Matching

Edge	Node Weight						
(u, v)	u	v	u-Avg(u)	v-Avg(v)	[u-Avg(u)]^2	[v-Avg(v)]^2	[u-Avg(u)][v-Avg(v)]
[1-3]	0.5	0.9	0.07	0.23	0.0049	0.0529	0.0161
[2-4]	0.7	0.3	0.27	-0.37	0.0729	0.1369	-0.0999
[5-6]	0.1	0.8	-0.57	0.13	0.3249	0.0169	-0.0741
Avg	0.43	0.67		Sum	0.4027	0.2067	-0.1579

$$\text{Assortativity Index} = \frac{\text{Sum}\{[u\text{-Avg}(u)][v\text{-Avg}(v)]\}}{\text{Sqrt}(\text{Sum}\{[u\text{-Avg}(u)]^2\}) * \text{Sqrt}(\text{Sum}\{[v\text{-Avg}(v)]^2\})}$$

$$\text{Assortativity Index} = \frac{0.0161 – 0.0999 – 0.0741}{\text{Sqrt}(0.4027) * \text{Sqrt}(0.2067)}$$

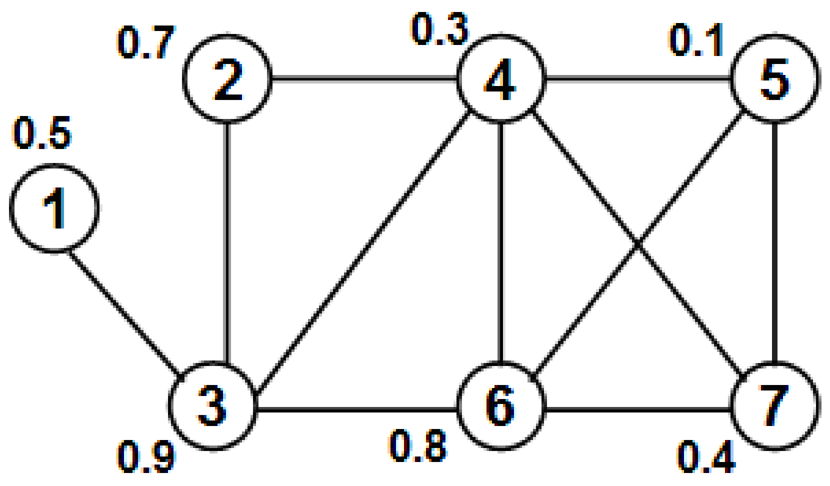
Assortativity Index = -0.55 % Node Matches = (3*2)/7 = 86%

Maximal Assortative Matching (MAM) and Maximal Dissortative Matching (MDM)

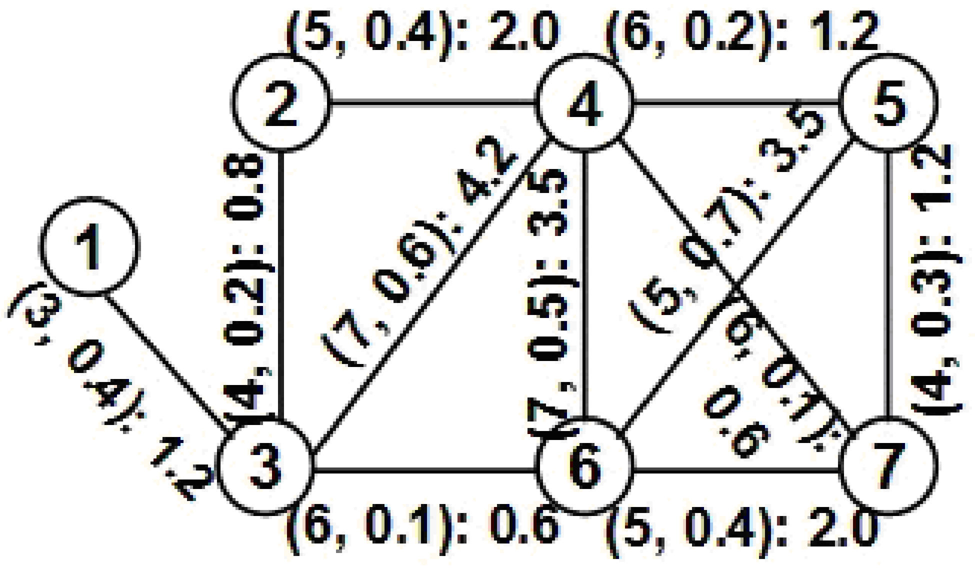
- We know that the assortative index of a set of edges can be from -1 to 1.
- MAM: A maximal matching whose assortative index is as large as possible (close to 1)
  - Maximal matching of similar vertices
- MDM: A maximal matching whose assortative index is as small as possible (close to -1)
  - Maximal matching of dissimilar vertices
- **Assortative Weight of an Edge:**
  - **Coverage Weight * Absolute value of the difference in the weights of the end vertices of the edge**
- **MAM Algorithm:** Run the MNM algorithm by removing the edge with the smallest assortativity weight in each iteration.
- **MDM Algorithm:** Run the MNM algorithm by removing the edge with the largest assortativity weight in each iteration.

N. Meghanathan, "Maximal Assortative Matching and Maximal Dissortative Matching for Complex Network Graphs," accepted for publication in *The Computer Journal*, 2016. DOI: 10.1093/comjnl/bxv102

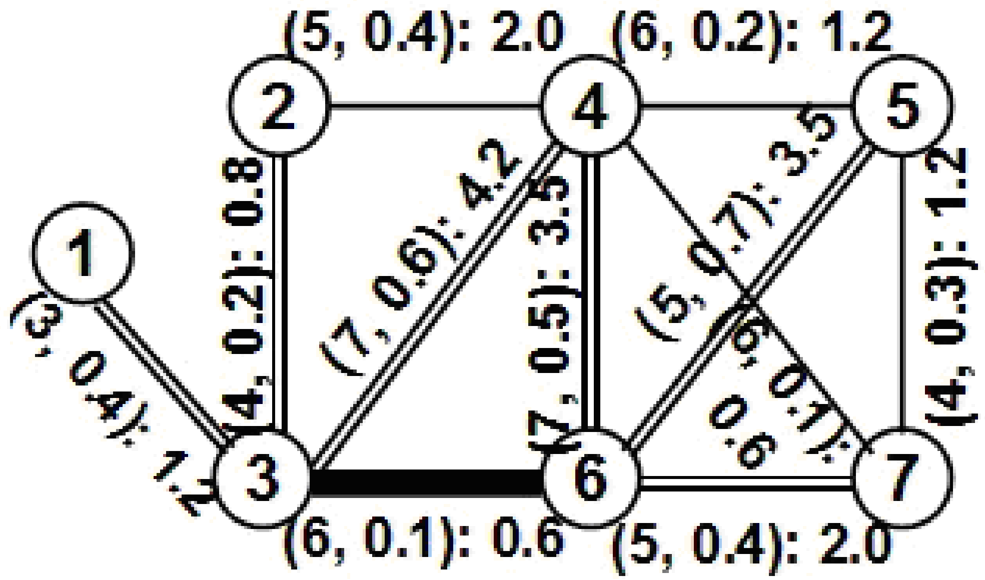
MAM Example 1



Given Graph

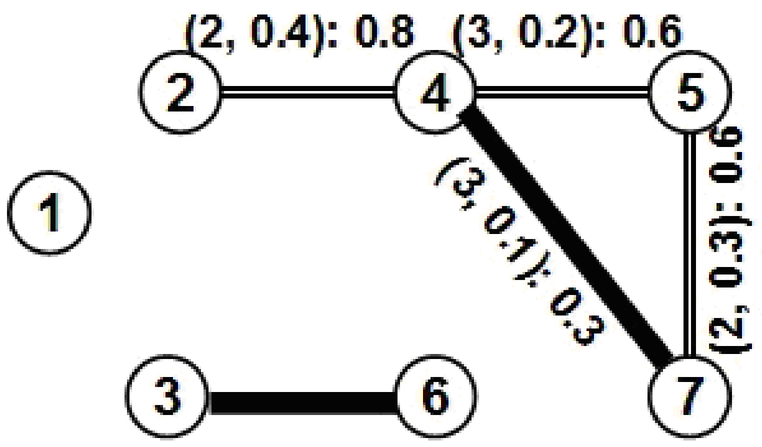


Initialization: Assortative Edge Weights

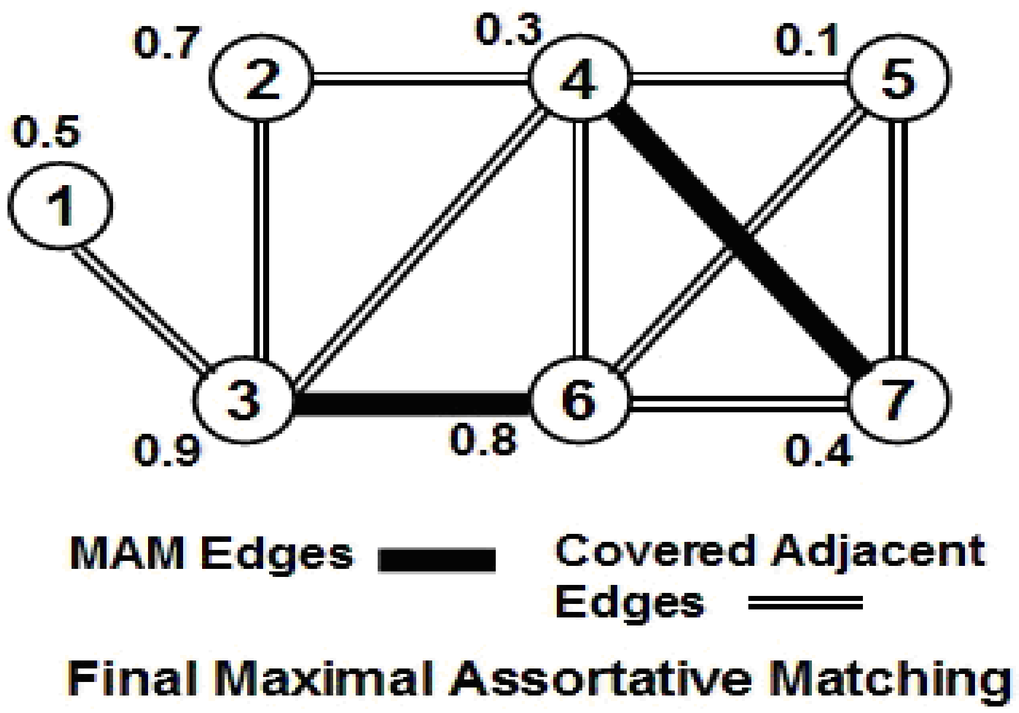


Iteration 1
Edge 3 – 6 is chosen for MAM
All its adjacent edges are removed.

MAM Example 1 (2)



Iteration 2
Edge 4 – 7 is chosen for MAM
All its adjacent edges are removed.



MAM Example 1 (3)

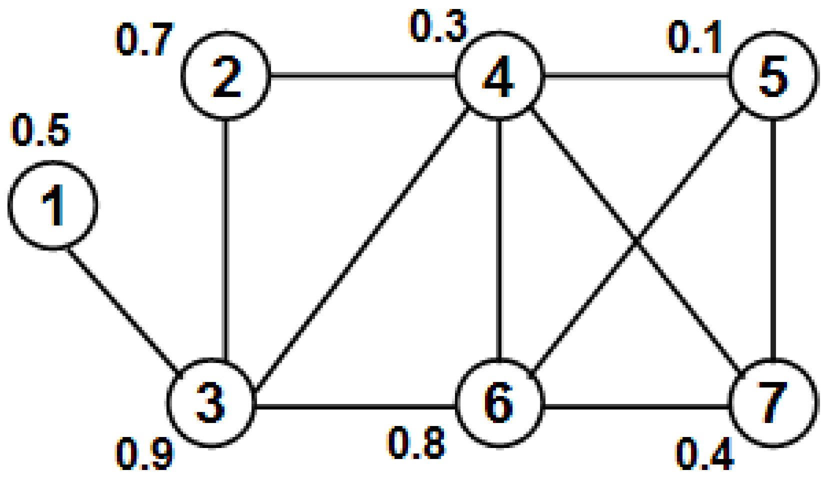
Edge	Node Weight						
(u, v)	u	v	u-Avg(u)	v-Avg(v)	$[u\text{-Avg}(u)]^2$	$[v\text{-Avg}(v)]^2$	[u-Avg(u)][v-Avg(v)]
[3-6]	0.9	0.8	0.3	0.2	0.09	0.04	0.06
[4-7]	0.3	0.4	-0.3	-0.2	0.09	0.04	0.06
Avg	0.6	0.6		Sum	0.18	0.08	0.12

$$\text{Assortativity Index} = \frac{\text{Sum}\{[u\text{-Avg}(u)][v\text{-Avg}(v)]\}}{\text{Sqrt}(\text{Sum}\{[u\text{-Avg}(u)]^2\}) * \text{Sqrt}(\text{Sum}\{[v\text{-Avg}(v)]^2\})}$$

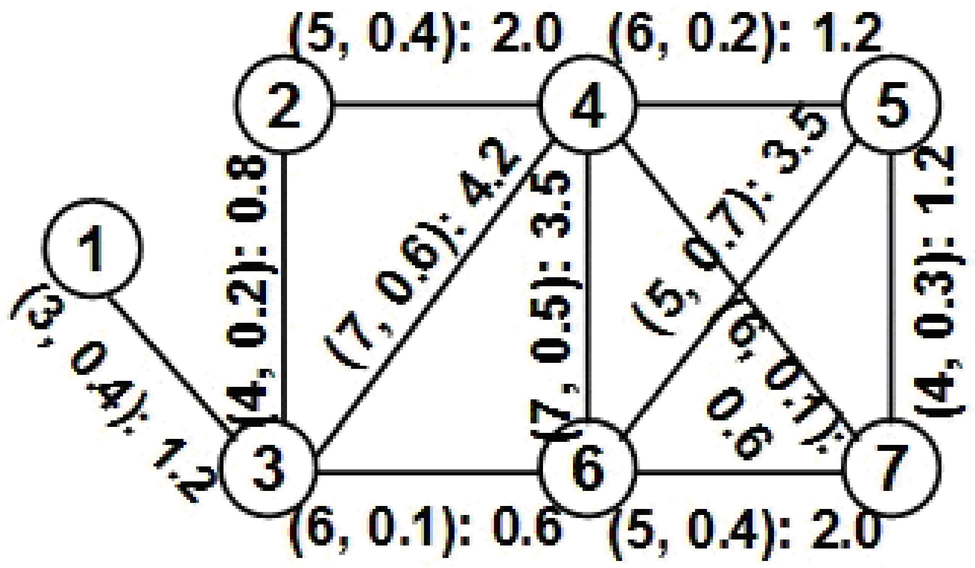
$$\text{Assortativity Index} = \frac{0.06 + 0.06}{\text{Sqrt}(0.09+0.08) * \text{Sqrt}(0.04+0.04)}$$

Assortativity Index = 1.0 % Node Matches = (2*2)/7 = 57%

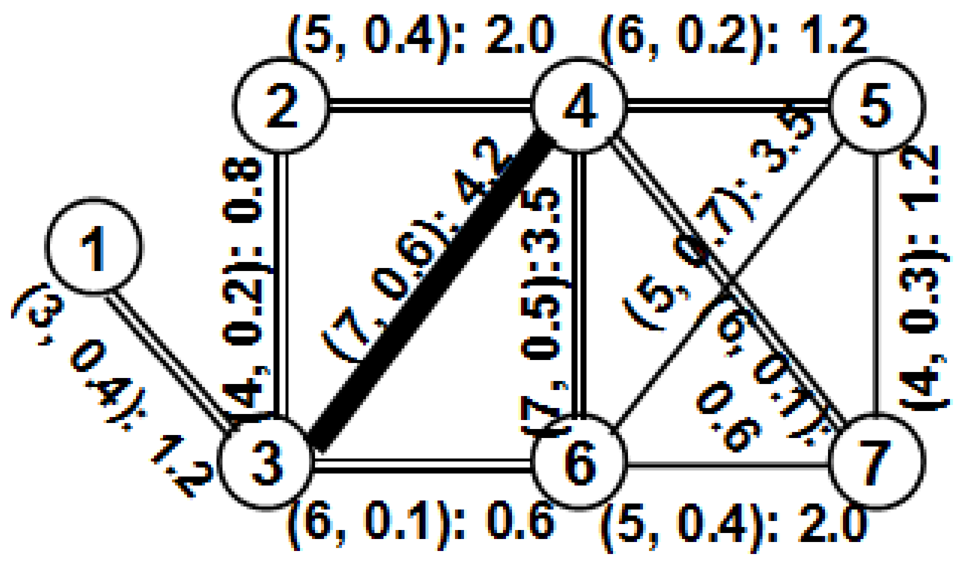
MDM Example 1



Given Graph

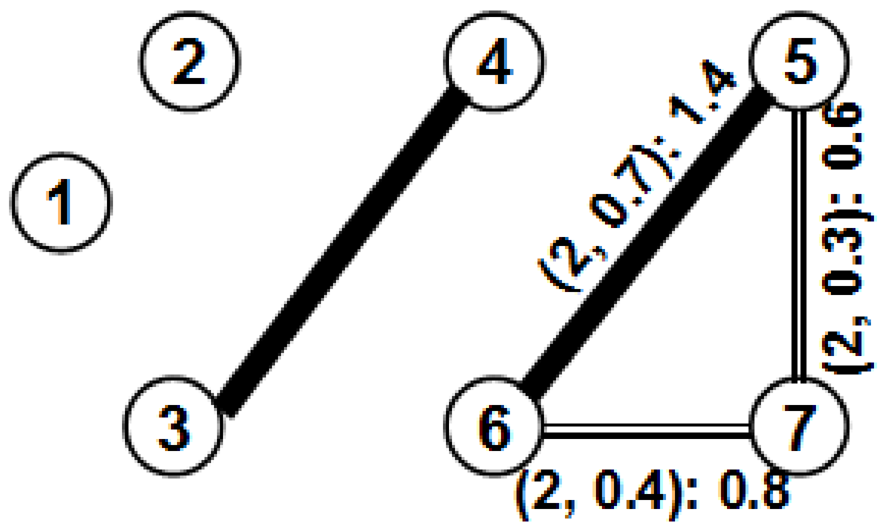


Initialization: Assortative Edge Weights

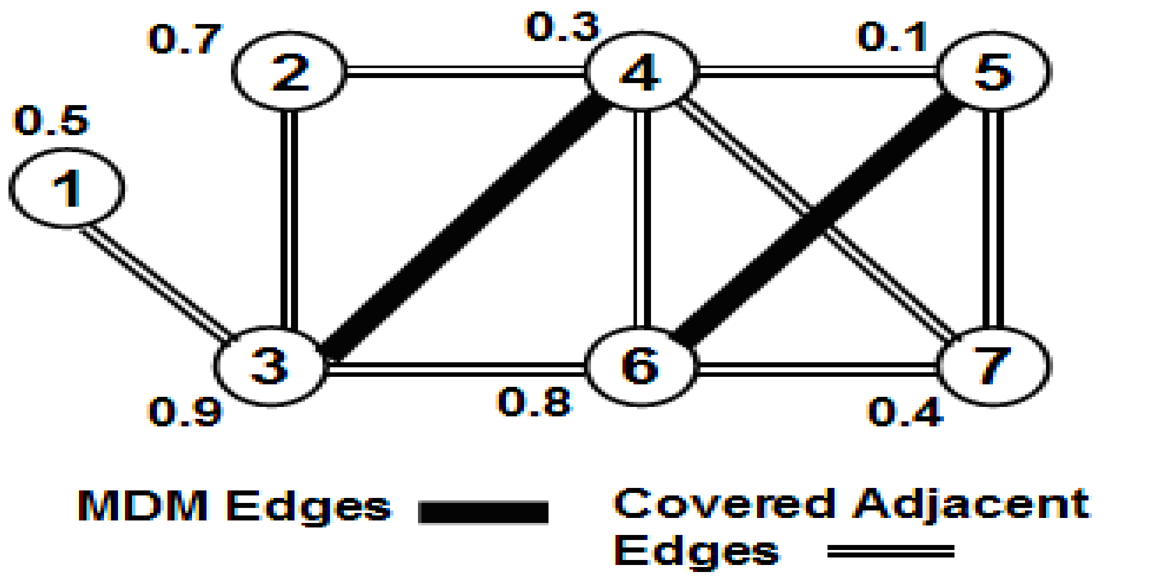


Iteration 1
Edge 3 – 4 is chosen for MDM
All its adjacent edges are removed.

MDM Example 1 (2)



Iteration 2
Edge 5 – 6 is chosen for MDM
All its adjacent edges are removed.



Final Maximal Dissortative Matching

MDM Example 1 (3)

Edge	Node Weight						
(u, v)	u	v	u-Avg(u)	v-Avg(v)	$[u\text{-Avg}(u)]^2$	$[v\text{-Avg}(v)]^2$	[u-Avg(u)][v-Avg(v)]
[3-4]	0.9	0.3	0.4	-0.25	0.16	0.0625	-0.10
[5-6]	0.1	0.8	-0.4	0.25	0.16	0.0625	-0.10
Avg	0.5	0.55		Sum	0.32	0.125	-0.20

$$\text{Assortativity Index} = \frac{\text{Sum}\{[u\text{-Avg}(u)][v\text{-Avg}(v)]\}}{\text{Sqrt}(\text{Sum}\{[u\text{-Avg}(u)]^2\}) * \text{Sqrt}(\text{Sum}\{[v\text{-Avg}(v)]^2\})}$$

$$\text{Assortativity Index} = \frac{-0.20}{\text{Sqrt}(0.16+0.16) * \text{Sqrt}(0.0625+0.0625)}$$

Assortativity Index = -1.0 % Node Matches = (2*2)/7 = 57%

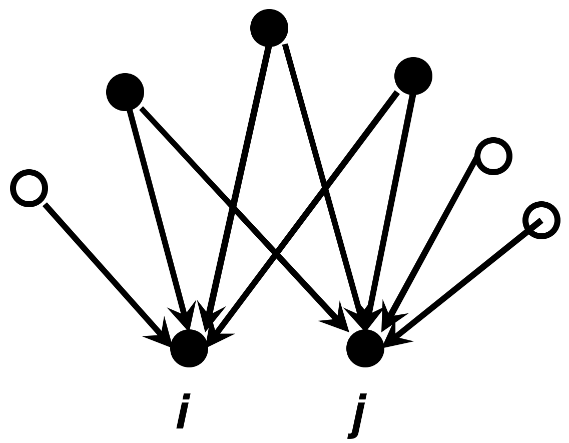
Cocitation and Bibliographic Coupling

- The CB-Adjacency matrix is the one where there is a 1 in (row index i, column index j) if there is an edge j to i.
  - Aij = 1 iff there is an edge j → I
  - Aij = 0 iff there is NO edge from j to i.
- Cocitation and Bibliographic coupling are some of the techniques to transform a directed graph to an undirected graph and analyze the info hidden in the directed graph.
- The <u>**cocitation of two vertices**</u> i and j in a directed graph is the number of vertices k that have outgoing edges pointing to both i and j.
  - Cocitation $C_{ij} = 1$ iff $A_{ik} = 1$ and $A_{jk} = 1$.

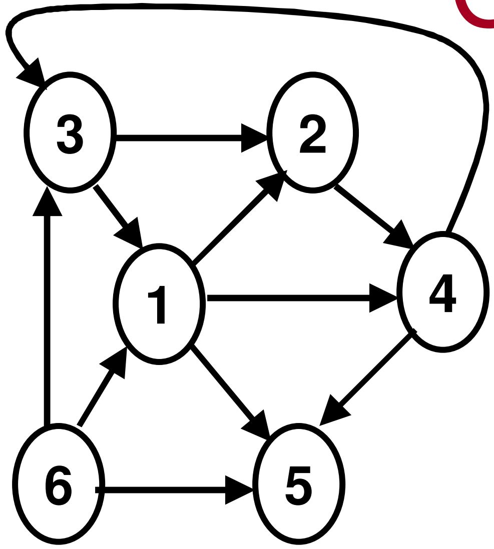
$$C_{ij} = \sum_{k=1}^{n} A_{ik} A_{jk} = \sum_{k=1}^{n} A_{ik} A_{kj}^{T}$$

$$\mathbf{C} = \mathbf{A}\mathbf{A}^{\mathrm{T}}$$

$\mathbf{C}_{ij} = \mathbf{3}$



Cocitation Coupling: Example



CB Adj. Matrix A =

$$\left(\begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 2 & 1 & 0 & 1 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 1 & 0 & 1 \\ 4 & 1 & 1 & 0 & 0 & 0 & 0 \\ 5 & 1 & 0 & 0 & 1 & 0 & 1 \\ 6 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right) \quad A^T = \left(\begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 2 & 0 & 0 & 0 & 1 & 0 & 0 \\ 3 & 1 & 1 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 1 & 0 & 1 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 6 & 1 & 0 & 1 & 0 & 1 & 0 \end{array}\right)$$

Cocitation Coupling Matrix =

$$\left(\begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 2 & 1 & 0 & 1 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 1 & 0 & 1 \\ 4 & 1 & 1 & 0 & 0 & 0 & 0 \\ 5 & 1 & 0 & 0 & 1 & 0 & 1 \\ 6 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right) \left(\begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 2 & 0 & 0 & 0 & 1 & 0 & 0 \\ 3 & 1 & 1 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 1 & 0 & 1 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 6 & 1 & 0 & 1 & 0 & 1 & 0 \end{array}\right) = \left(\begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 2 & 1 & 1 & 0 & 1 & 0 \\ 2 & 1 & 2 & 0 & 1 & 1 & 0 \\ 3 & 1 & 0 & 2 & 0 & 2 & 0 \\ 4 & 0 & 1 & 0 & 2 & 1 & 0 \\ 5 & 1 & 1 & 2 & 1 & 3 & 0 \\ 6 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right)$$

Other than the entries for a vertex to itself, the only entries where Cij > 1 are: C35 = C53 = 2; meaning that two papers (4 and 6) are citing both papers 3 and 5.

Cocitation Coupling

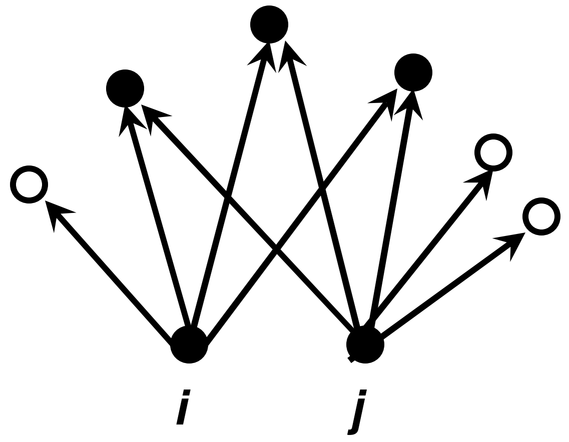
- A cocitation network comprises of only undirected edges (i, j), iff $C_{ij} > 0$.
- The value of C_{ij} is a good indicator of two papers i and j that deal with related topics.
  - If two papers are often cited together in the same bibliography, they probably have something in common.
  - The more often they are cited together, the more likely it is that they are related.
- **Strength:** Cocitation counts of papers increase with time. The rate of increase can be used to trace the evolution of an academic field.
- The co-citation measure reflects the opinion of many authors.
- **Weakness with Cocitation coupling**: The relative similarity between two papers is being adjudged with the number of papers citing them.
- For two papers i and j to be adjudged to be "strongly related" to each other, we should have more incoming edges to both of them.
  - This may not be the case for two papers (or at least one of them) that have few citations.
  - Also, the relative similarity of two papers cannot be computed until both the papers are cited by at least one paper.

Bibliographic Coupling

- Two papers i and j are related if they refer to one or more papers k in common.
  - The number of common references is an indicator of the similarity between the two papers.
    - However, the similarity is based on the opinion of only the authors of the two papers; not others in the subject area – a weakness to assess similarity between two papers.
  - It is a static measure: established when a paper gets published and not updated henceforth.
    - Hence, it cannot be used to trace the evolution of an academic field.
  - Strength: Unlike Co-citation coupling, there is no need to wait for other papers to cite.

Bibliographic Coupling

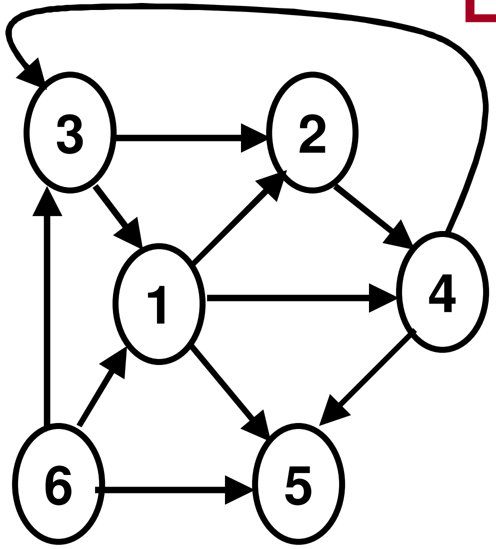
- The **bibliographic coupling of two vertices** i and j in a directed graph is the number of vertices k that have incoming edges from both i and j.
  - Bibliographic coupling B_{ij} = 1 iff $A_{ki} = 1$ and $A_{kj} = 1$.



$$\boxed{B_{ij} = \sum_{k=1}^{n} A_{ki} A_{kj} = \sum_{k=1}^{n} A^{T}_{ik} A_{kj}}$$

$$\boxed{\mathrm{B} = \mathrm{A}^{\mathrm{T}}\mathrm{A}}$$

Bibliographic Coupling: Example



$$\text{CB Adj. Matrix } A = \left(\begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 2 & 1 & 0 & 1 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 1 & 0 & 1 \\ 4 & 1 & 1 & 0 & 0 & 0 & 0 \\ 5 & 1 & 0 & 0 & 1 & 0 & 1 \\ 6 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right) \quad A^T = \left(\begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 2 & 0 & 0 & 0 & 1 & 0 & 0 \\ 3 & 1 & 1 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 1 & 0 & 1 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 6 & 1 & 0 & 1 & 0 & 1 & 0 \end{array}\right)$$

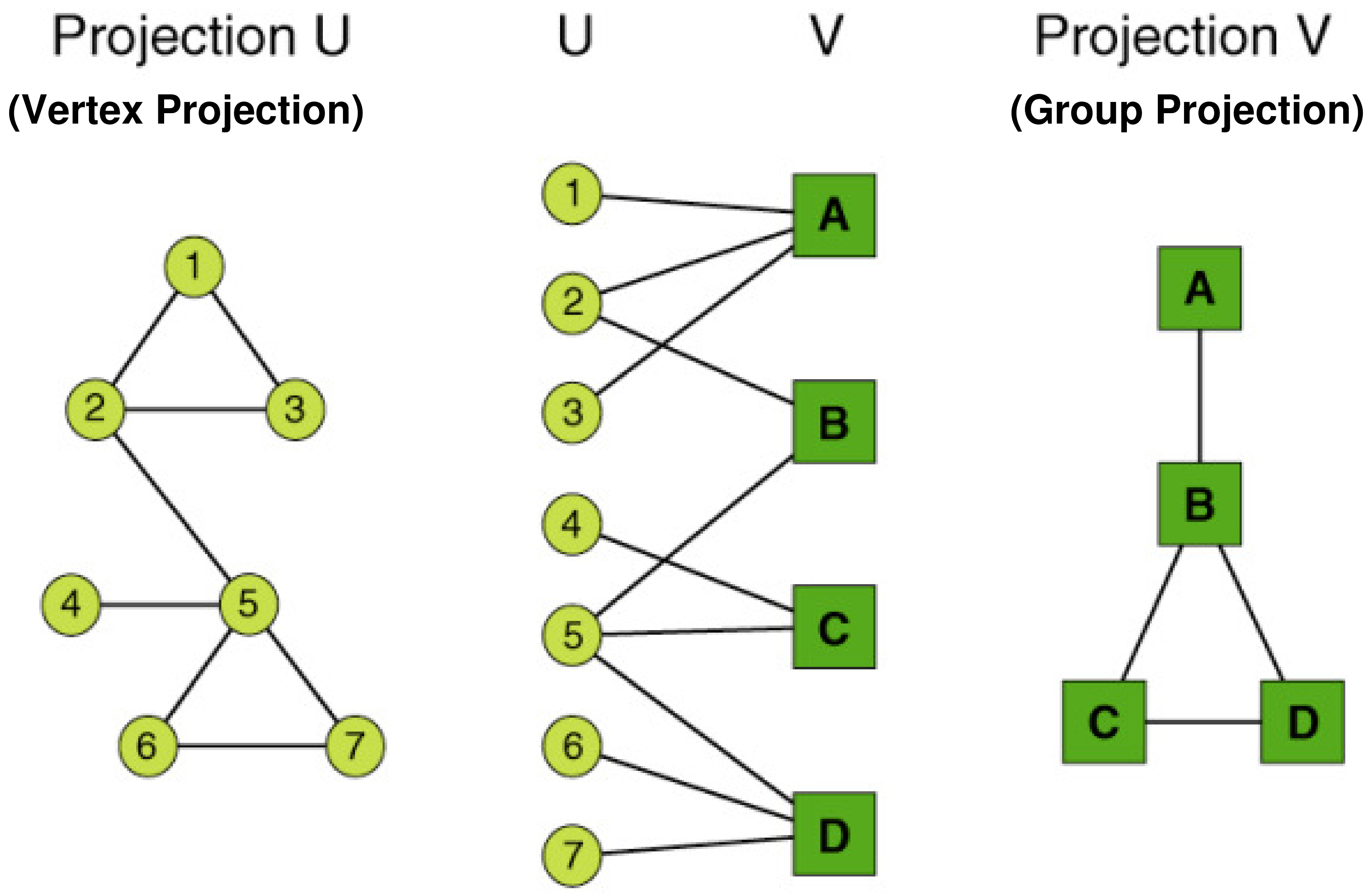
$$\text{Bibliogr. Coupling Matrix} = \left(\begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 2 & 0 & 0 & 0 & 1 & 0 & 0 \\ 3 & 1 & 1 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 1 & 0 & 1 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 6 & 1 & 0 & 1 & 0 & 1 & 0 \end{array}\right) \left(\begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 2 & 1 & 0 & 1 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 1 & 0 & 1 \\ 4 & 1 & 1 & 0 & 0 & 0 & 0 \\ 5 & 1 & 0 & 0 & 1 & 0 & 1 \\ 6 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right) = \left(\begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 3 & 1 & 1 & 1 & 0 & 1 \\ 2 & 1 & 1 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & 2 & 0 & 0 & 1 \\ 4 & 1 & 0 & 0 & 2 & 0 & 2 \\ 5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 6 & 1 & 0 & 1 & 2 & 0 & 3 \end{array}\right)$$

Other than the entries for a vertex to itself, the only entries where Bij > 1 are: B46 = B64 = 2; meaning that two papers (1 and 3) are both being referred by papers 4 and 6.

Bipartite Graph and Projection

- A bipartite graph (or bigraph) is a network whose nodes are divided into two disjoint sets U and V; the only links in the graph are those connecting a U-node to a V-node.
  - There is no link connecting two U-nodes or two V-nodes.
  - The U-nodes can be of one color and the V-nodes can be of another color; a link always connects two nodes of different colors.
- Projection:
  - Projection U: Links involving the U-nodes. Two U-nodes are connected if they link to the same V-node in the bipartite graph.
  - Projection V: Links involving the V-nodes. Two V-nodes are connected if they link to the same U-node in the bipartite graph.

Bipartite Graph and Projection



Source: Fig. 2.9a, Barabasi

Incidence Matrix and Projections

Vertex Projection
Two vertices are connected if they belong to at least one common group.

$$VP_{ij} = \sum_{k=1}^{g} B_{ki} B_{kj}$$

$$VP_{ij} = \sum_{k=1}^{g} B_{ik}^{T} B_{kj}$$

$$\mathrm{VP} = \mathrm{B}^{\mathrm{T}}\mathrm{B}$$

VP_{ij} is the number of groups that i and j share.
VP_{ii} is the number of groups to which i belongs to.

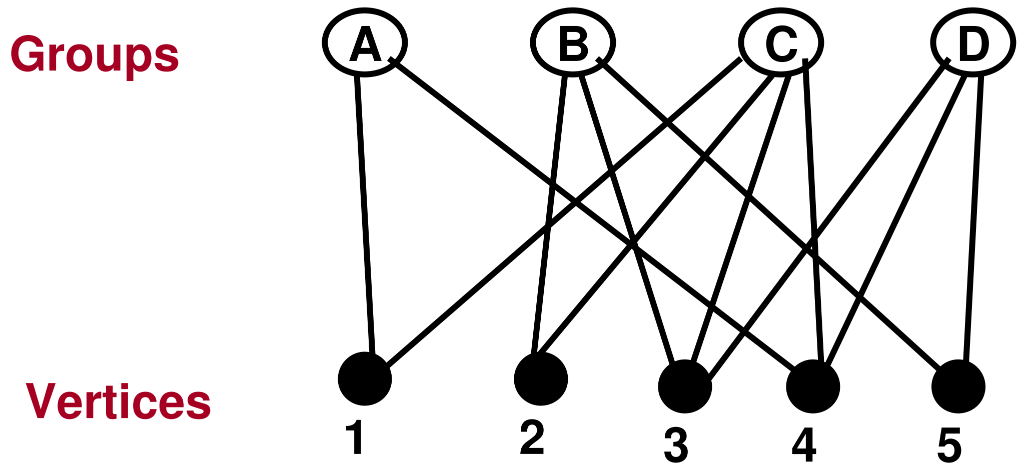
Group Projection
Two groups are connected if they share at least one common vertex.

$$GP_{ij} = \sum_{k=1}^{n} B_{ik} B_{jk}$$

$$GP_{ij} = \sum_{k=1}^{n} B_{ik} B_{kj}^{T}$$

$$\mathrm{GP} = \mathrm{B}\mathrm{B}^{\mathrm{T}}$$

GP_{ij} is the number of vertices that groups i and j share.
GP_{ii} is the number of vertices that belong to group i.



$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline A & 1 & 0 & 0 & 1 & 0 \\ B & 0 & 1 & 1 & 0 & 1 \\ C & 1 & 1 & 1 & 1 & 0 \\ D & 0 & 0 & 1 & 1 & 1 \end{array}$$

Adjacency Matrix B

$$\underbrace{\boxed{\begin{array}{ccccc} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{array}}}_{\mathbf{B}} \; \underbrace{\boxed{\begin{array}{cccc} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{array}}}_{\mathbf{B^T}} = \underbrace{\begin{array}{c|cccc} & A & B & C & D \\ \hline A & 2 & 0 & 2 & 1 \\ B & 0 & 3 & 2 & 2 \\ C & 2 & 2 & 4 & 2 \\ D & 1 & 2 & 2 & 3 \end{array}}_{\mathbf{BB^T}}$$

Group Projection: Indicates the number of vertices that are common to any two groups.

$$\underbrace{\boxed{\begin{array}{cccc} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{array}}}_{\mathbf{B^T}} \; \underbrace{\boxed{\begin{array}{ccccc} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{array}}}_{\mathbf{B}} = \underbrace{\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 2 & 1 & 1 & 2 & 0 \\ 2 & 1 & 2 & 2 & 1 & 1 \\ 3 & 1 & 2 & 3 & 2 & 2 \\ 4 & 2 & 1 & 2 & 3 & 1 \\ 5 & 0 & 1 & 2 & 1 & 2 \end{array}}_{\mathbf{B^TB}}$$

Vertex Projection: Indicates the number of common groups for any two vertices.

Examples of Bipartite Graphs and Projections

- Actor-movie network: Actors are one set of nodes and the movies are another set of nodes. Each actor is connected to the movie(s) in which s/he has acted.
  - Projection Actors (Actor network): Two actors are connected if they acted together in at least one movie
  - Projection Movies (Movie network): Two movies are connected if they had at least one common actor.
- Diseasome network: One set of nodes are the diseases and another set of nodes are the genes: A disease is connected to a gene if mutations in that gene are known to affect the particular disease.
  - Projection Gene (Gene network): Two genes are connected if they are associated with the same disease.
  - Projection Disease (Disease network): Two diseases are connected if the same genes are associated with them, indicating the two diseases have common genetic origins.

Paths and Distances in Networks

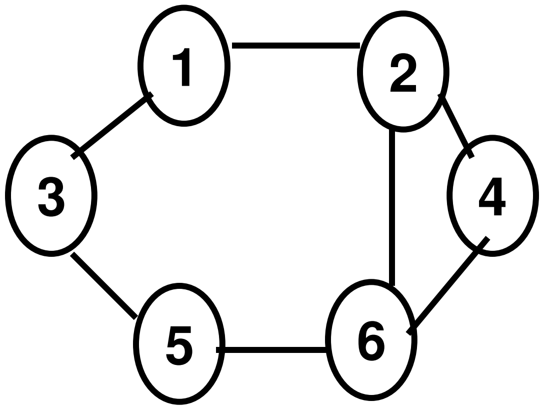
- A path between two nodes *i* and *j* is a route along the links of the network; the length (distance d_{ij}) is the number of links the path contains.
  - In an undirected network, $d_{ij} = d_{ji}$
  - In a directed network, d_{ij} need not be equal to d_{ji}
- Shortest path (geodesic path): between any two nodes i and j is the path with the fewest number of links.
- Diameter of a network: Maximum of the shortest path lengths between any two nodes
- The number of paths (walks) of length *k* between any two vertices can be found from: A^k where A is the adjacency matrix of the network.
- The shortest path length between any two nodes i and j is the minimum value of *k* for which $A^{k-1}[i, j] = 0$ and $A^k[i, j] > 0$.

Eccentricity, Diameter, Radius, Center

- The eccentricity of a node is the maximum of the shortest path distance (# hops) to any other node in the network.
- The diameter of the network is the maximum of the node eccentricity values.
- The radius of the network is the minimum of the node eccentricity values.
- A node is said to be central if its eccentricity is equal to the radius of the network.
- The set of nodes that are central constitute the center of the network.
- **Weiner Index:**

$$\boxed{W(G) = \sum_{u=1}^{n}\sum_{v=1}^{n} dist(u,v)}$$

Average Path Length: $\boxed{\dfrac{W(G)}{n(n-1)}}$

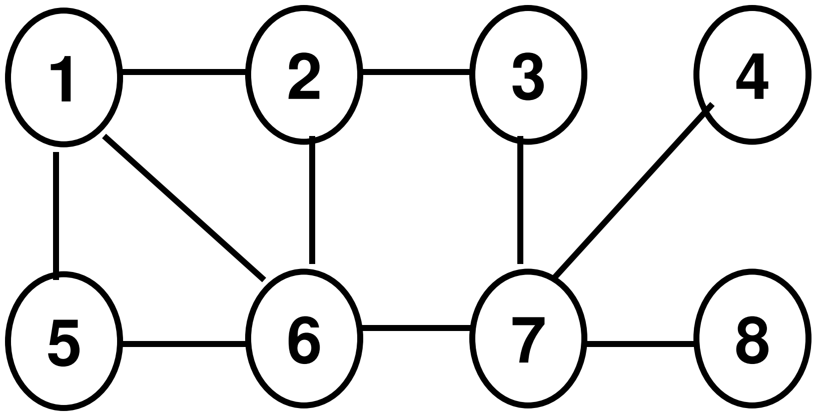


Nodes	**Distances (Eccentricity)**
1	1, 1, 2, 2, 2 (2)
2	1, 1, 1, 2, 2 (2)
3	1, 1, 2, 2, 3 (3)
4	1, 1, 2, 2, 3 (3)
5	1, 1, 2, 2, 2 (2)
6	1, 1, 1, 2, 2 (2)

Diameter = 3
Radius 2
Center = {1, 2, 5, 6}

Weiner Index = 48
Avg. Path Length = 48/ (6*5)
= 1.60

Example 2: Path Length, Diameter



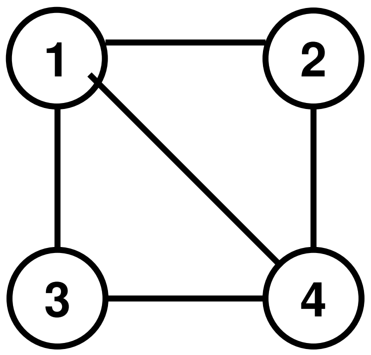
Diameter = 3
Radius 2
Center = {6, 7}

Weiner Index = 106
Avg. Path Length = 106/(8*7)
= 1.89

Distance Matrix

	1	2	3	4	5	6	7	8	Eccentricity	Sum
1	0	1	2	3	1	1	2	3	3	13
2	1	0	1	3	2	1	2	3	3	13
3	2	1	0	2	3	2	1	2	3	13
4	3	3	2	0	3	2	1	2	3	16
5	1	2	3	3	0	1	2	3	3	15
6	1	1	2	2	1	0	1	2	2	10
7	2	2	1	1	2	1	0	1	2	10
8	3	3	2	2	3	2	1	0	3	16

# Walks (Paths) of Certain Length



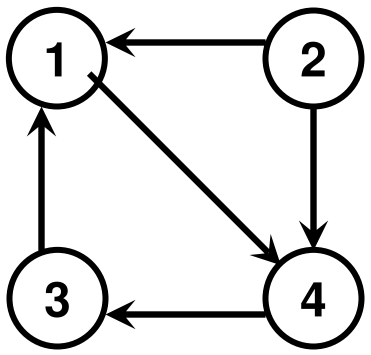
$$\begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 1 & 1 & 1 \\ 2 & 1 & 0 & 0 & 1 \\ 3 & 1 & 0 & 0 & 1 \\ 4 & 1 & 1 & 1 & 0 \end{array}$$

A Walk is a sequence of vertices connecting source and destination such that any vertex (including the end vertices) could occur even more than once. In a path, an intermediate vertex (if any is present) could occur only once.

Path: 2 – 1 – 3 **Length: 2**
Walk: 2 – 1 – 2 – 4 – 3 **Length: 4**

$$A^2 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 1 & 1 & 1 \\ 2 & 1 & 0 & 0 & 1 \\ 3 & 1 & 0 & 0 & 1 \\ 4 & 1 & 1 & 1 & 0 \end{array} \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 1 & 1 & 1 \\ 2 & 1 & 0 & 0 & 1 \\ 3 & 1 & 0 & 0 & 1 \\ 4 & 1 & 1 & 1 & 0 \end{array} = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 3 & 1 & 1 & 2 \\ 2 & 1 & 2 & 2 & 1 \\ 3 & 1 & 2 & 2 & 1 \\ 4 & 2 & 1 & 1 & 3 \end{array}$$

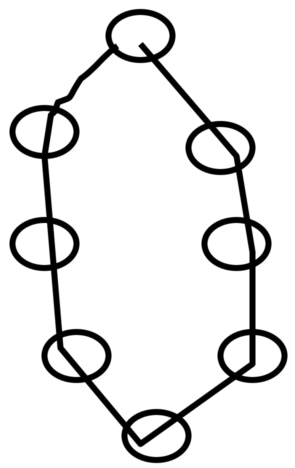
Number of Paths of Certain Length



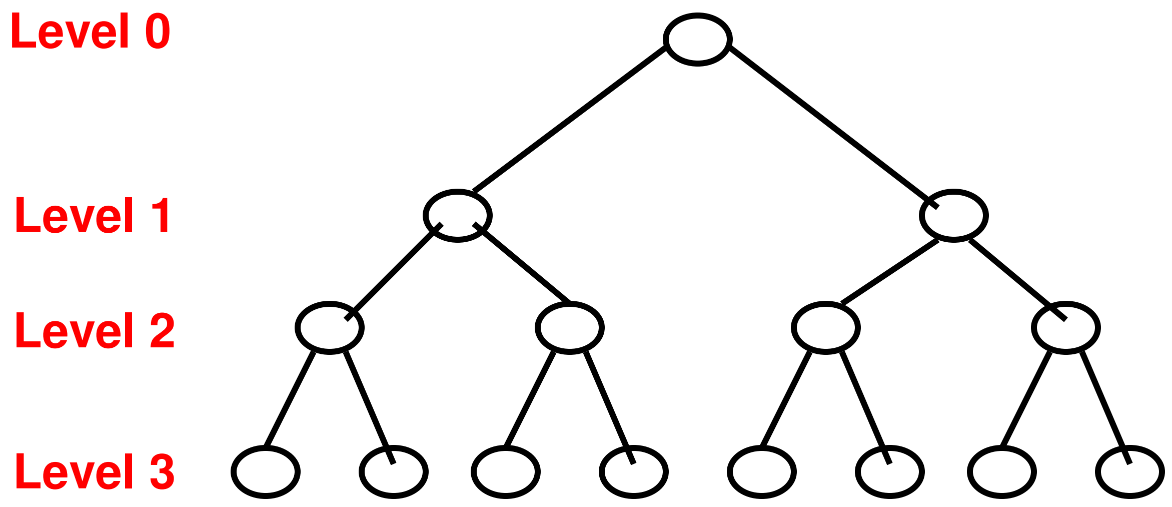
$$\begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 0 & 0 & 1 \\ 2 & 1 & 0 & 0 & 1 \\ 3 & 1 & 0 & 0 & 0 \\ 4 & 0 & 0 & 1 & 0 \end{array}$$

$$A^2 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 0 & 0 & 1 \\ 2 & 1 & 0 & 0 & 1 \\ 3 & 1 & 0 & 0 & 0 \\ 4 & 0 & 0 & 1 & 0 \end{array} \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 0 & 0 & 1 \\ 2 & 1 & 0 & 0 & 1 \\ 3 & 1 & 0 & 0 & 0 \\ 4 & 0 & 0 & 1 & 0 \end{array} = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 & 1 \\ 3 & 0 & 0 & 0 & 1 \\ 4 & 1 & 0 & 0 & 0 \end{array}$$

Diameter



Diameter of a Ring = N/2 or (N-1)/2

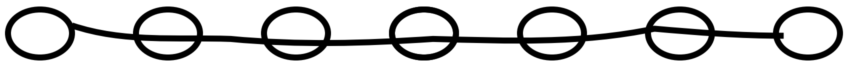


Diameter of a binary tree with K levels

# nodes N = $2^{(K+1)} - 1$

$K+1 = \log_2(N+1)$

Diameter = 2K = $2*[\log_2(N+1) - 1]$



Diameter of a Chain = (N-1)

Small Average Path Length and Diameter

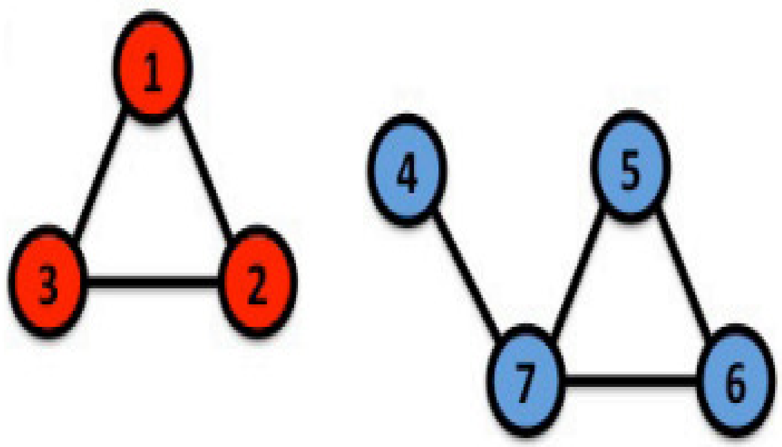
- Co-Authorship studies
  - Grossman (2002) Math mean 7.6, max 27
  - Newman (2001) Physics mean 5.9, max 20
  - Goyal et al (2004) Economics mean 9.5, max 29
- WWW
  - Adamic, Pitkow (1999) – mean 3.1 (85.4% possible of 50M pages)
- Facebook
  - Backstrom et al (2012) – mean 4.74 (721 million users)

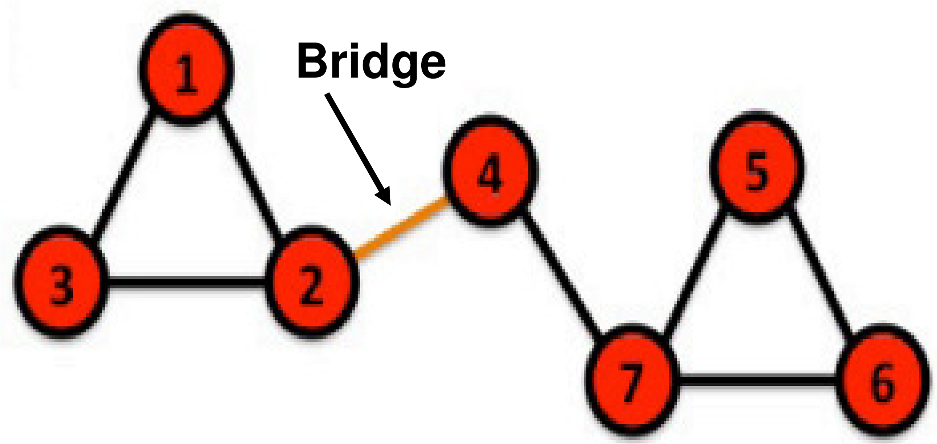
- <u>Small-World Property:</u>
- A network of *n* nodes is said to exhibit the “small-world” property if the average path length of the network is proportional to ln(n)
  - Observed for random networks with Poisson degree distribution
- <u>Ultra Small-World Property:</u>
- A network of *n* nodes is said to exhibit the “ultra small-world” property if the average path length of the network is proportional to ln(n)/ln(ln(n)).
  - Observed for scale-free networks with Power-law degree distribution

n = 100	**ln(n) = 4.61**	**ln(n)/ln(ln(n)) = 3.02**
n = 10,000	**ln(n) = 9.21**	**ln(n)/ln(ln(n)) = 4.15**
n = 10,000,000	**ln(n) = 16.12**	**ln(n)/ln(ln(n)) = 5.79**

Components (Clusters)

- The vertices of a graph are said to be in a single component if there is a path between the vertices.
- A graph is said to be connected if all its vertices are in one single component; otherwise, the graph is said to be disconnected and consists of multiple components.
  - Adding one or more links (bridges) can connect the different components





Breadth First Search (BFS)

- BFS is a graph traversal algorithm (like DFS); but, BFS proceeds in a concentric breadth-wise manner (not depth wise) by first visiting all the vertices that are adjacent to a starting vertex, then all unvisited vertices that are two edges apart from it, and so on.
  - The above traversal strategy of BFS makes it ideal for determining minimum-edge (i.e., minimum-hop paths) on graphs.
- If the underling graph is connected, then all the vertices of the graph should have been visited when BFS is started from a randomly chosen vertex.
  - If there still remains unvisited vertices, the graph is not connected and the algorithm has to restarted on an arbitrary vertex of another connected component of the graph.
- BFS is typically implemented using a FIFO-queue (not a LIFO-stack like that of DFS).
  - The queue is initialized with the traversal's starting vertex, which is marked as visited. On each iteration, BFS identifies all unvisited vertices that are adjacent to the front vertex, marks them as visited, and adds them to the queue; after that, the front vertex is removed from the queue.
- When a vertex is visited for the first time, the corresponding edge that facilitated this visit is called the <u>*tree edge*</u>. When a vertex that is already visited is re-visited through a different edge, the corresponding edge is called a <u>*cross edge*</u>.

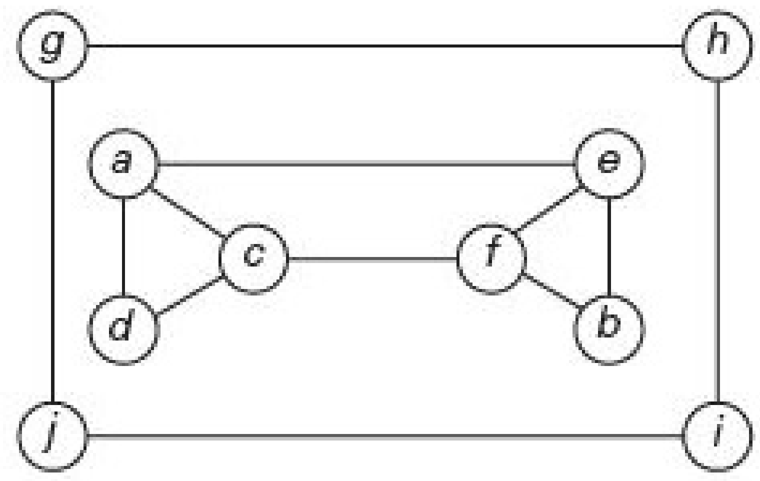
Pseudo Code of BFS

```
ALGORITHM  BFS(G)
    //Implements a breadth-first search traversal of a given graph
    //Input: Graph $G = \langle V, E \rangle$
    //Output: Graph $G$ with its vertices marked with consecutive integers
    //          in the order they are visited by the BFS traversal
    mark each vertex in $V$ with 0 as a mark of being "unvisited"
    $count \leftarrow 0$
    for each vertex $v$ in $V$ do
        if $v$ is marked with 0
            $bfs(v)$

    $bfs(v)$
    //visits all the unvisited vertices connected to vertex $v$
    //by a path and numbers them in the order they are visited
    //via global variable $count$
    $count \leftarrow count + 1$;   mark $v$ with $count$ and initialize a queue with $v$
    while the queue is not empty do
        for each vertex $w$ in $V$ adjacent to the front vertex do
            if $w$ is marked with 0
                $count \leftarrow count + 1$;   mark $w$ with $count$
                add $w$ to the queue
        remove the front vertex from the queue
```

BFS can be implemented with graphs represented as:
adjacency matrices: Θ(V^2); adjacency lists: Θ(|V|+|E|)

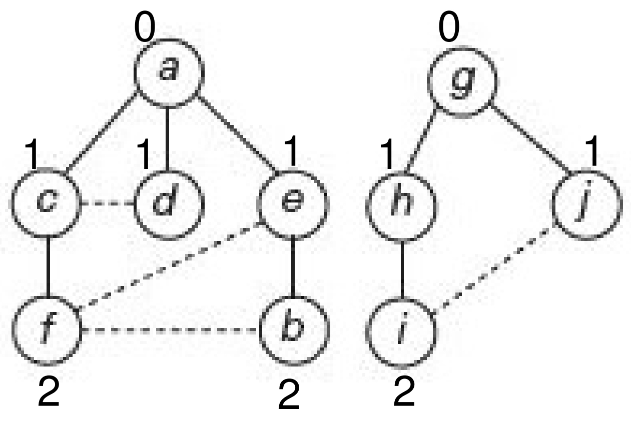
Example for BFS



(a)

$a_1\ c_2\ d_3\ e_4\ f_5\ b_6$
$g_7\ h_8\ j_9\ i_{10}$

(b)

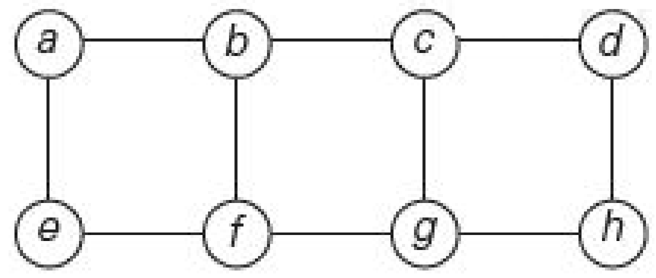


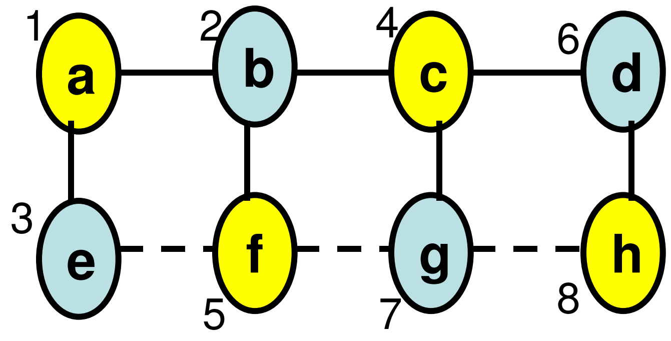
(c)

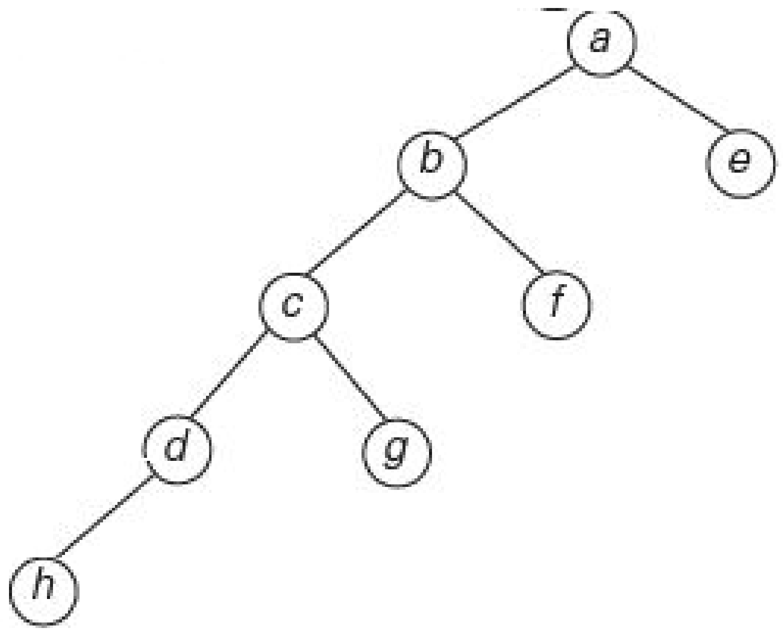
(a) Graph. (b) Traversal queue, with the numbers indicating the order in which the vertices are visited, i.e., added to (and removed from) the queue. (c) BFS forest with the tree and cross edges shown with solid and dotted lines, respectively.

Source: Figure 3.11: Levitin, 3rd Edition: Introduction to the Design and Analysis of Algorithms, 2012.

Use of BFS to find Minimum Edge Paths







Note: DFS cannot be used to find minimum edge paths, because DFS is not guaranteed to visit all the one-hop neighbors of a vertex, before visiting its two-hop neighbors and so on.

For example, if DFS is executed starting from vertex 'a' on the above graph, then vertex 'e' would be visited through the path a – b – c – d – h – g – f – e and not through the direct path a – e, available in the graph.

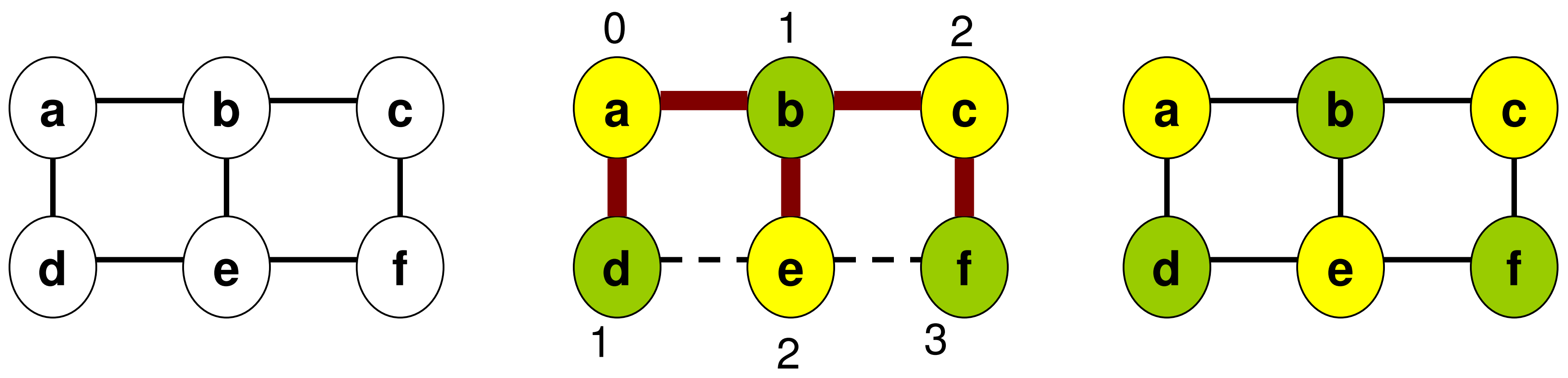
Source: Figure 3.12: Levitin, 3rd Edition: Introduction to the Design and Analysis of Algorithms, 2012.

Bi-Partite (2-Colorable) Graphs

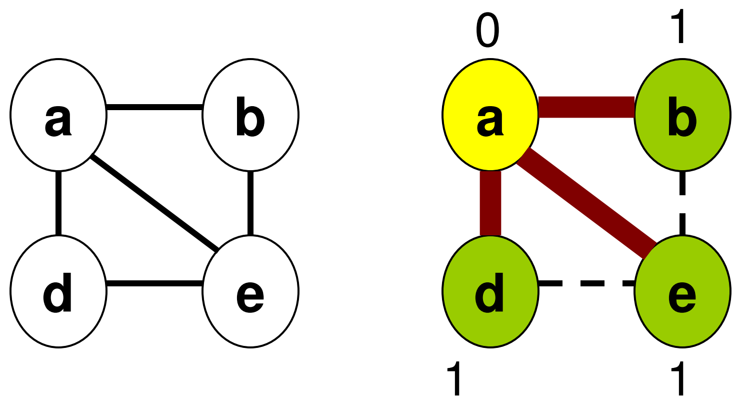
- A graph is said to be bi-partite or 2-colorable if the vertices of the graph can be colored in two colors such that every edge has its vertices in different colors.
- In other words, we can partition the set of vertices of a graph into two disjoint sets such that there is no edge between vertices in the same set. All the edges in the graph are between vertices from the two sets.
- We can check for the 2-colorable property of a graph by running a DFS or BFS
  - With BFS, if there are no cross-edges between vertices at the same level, then the graph is 2-colorable.
  - With DFS, if there are no back edges between vertices that are both at odd levels or both at even levels, then the graph is 2-colorable.
- We will use BFS as the algorithm to check for the 2-colorability of a graph.
  - The level of the root is 0 (consider 0 to be even).
  - The level of a child node is 1 more than the level of the parent node from which it was visited through a tree edge.
  - If the level of a node is even, then color the vertex in yellow.
  - If the level of a node is odd, then color the vertex in green.

Bi-Partite (2-Colorable) Graphs

Example for a 2-Colorable Graph

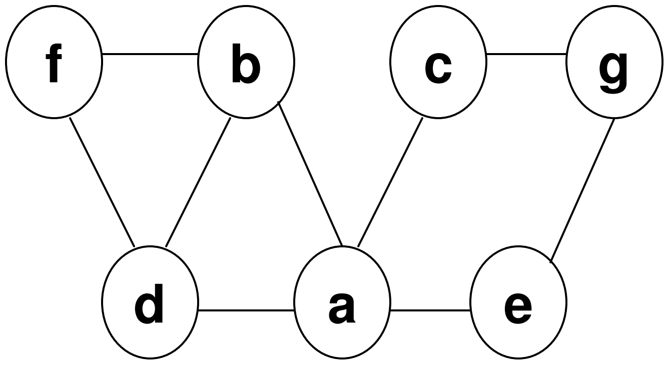


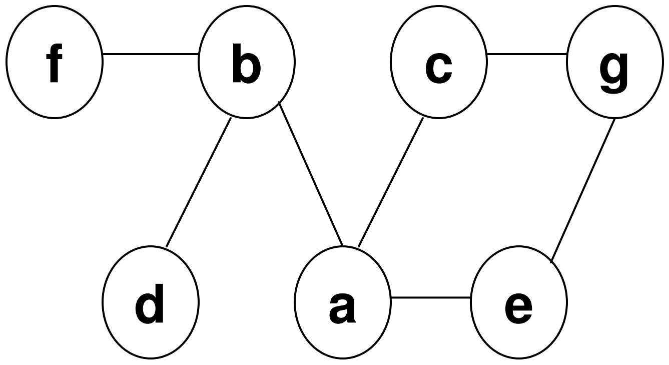
Example for a Graph that is Not 2-Colorable

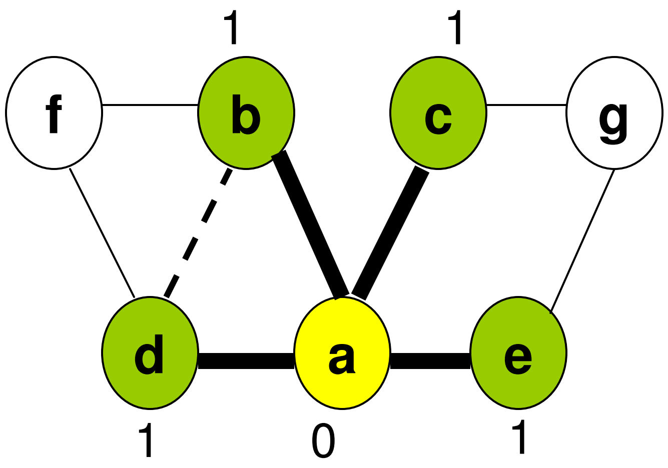


We encounter cross edges between vertices b and e; d and e – all the three vertices are in the same level.

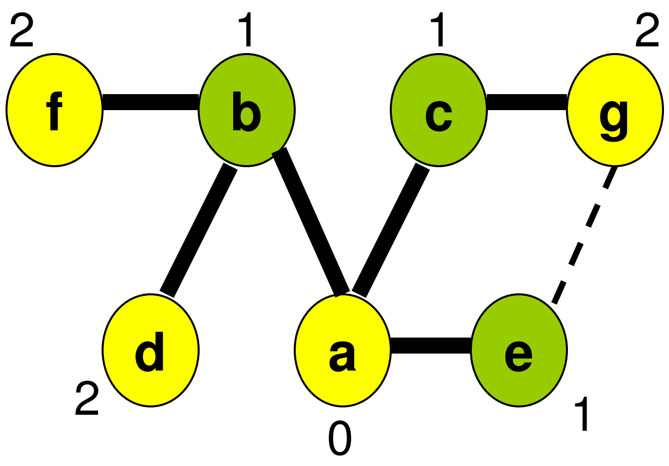
Examples: 2-Colorability of Graphs







b – d is a cross edge between Vertices at the same level. So, the graph is not 2-colorable

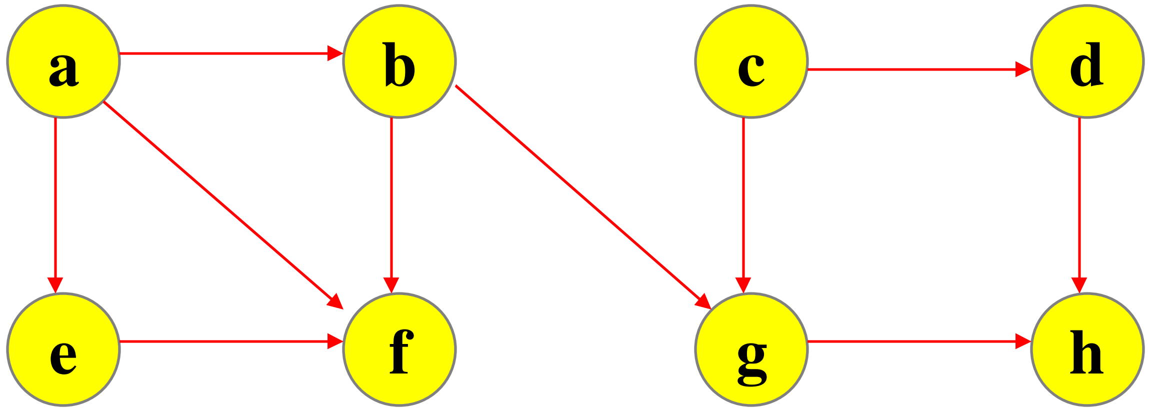


The above graph is 2-Colorable as there are no cross edges between vertices at the same level

Directed Acyclic Graphs (DAGs)

- A directed graph with no cycles.
  - E.g., Citation network: we always cite the work done earlier. A prior work does not cite a work to be done later.
- In a DAG, there must be at least one vertex that has only all incoming edges and no outgoing edge.
  - We start with one vertex, go around the vertices in the graph. If the path never reaches a vertex with no outgoing edges, then it must eventually arrive back at a vertex that has been visited previously – at most we can visit all the *n* vertices in the graph once before the path either terminates or we are forced to revisit a vertex (in the latter case, we encounter a cycle).
- To test for cycle: Remove the vertices with no outgoing edges (and all the associated incoming edges), one by one. If there is a cycle, there will be a scenario in which we could not find any vertex without outgoing edges.
  - If all vertices could be removed one by one, the graph is acyclic.

DFS on a DAG



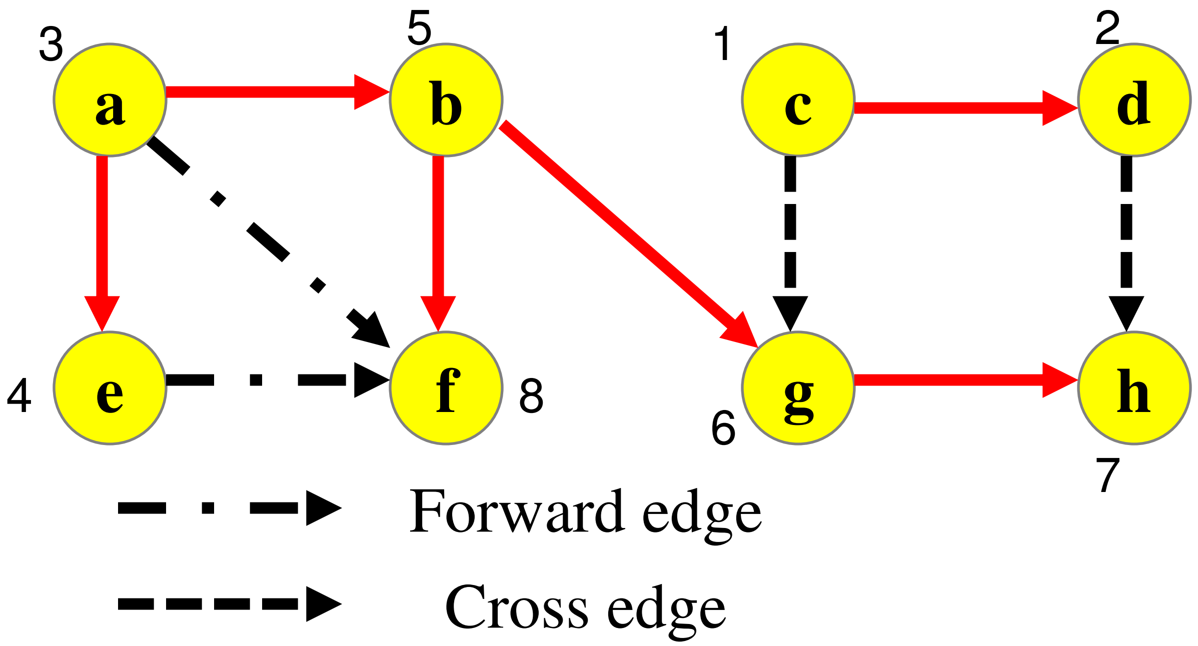
$h_{5,2}$

$g_{4,3}$

$f_{3,1}$

$b_{2,4}$ $e_{6,5}$ $d_{8,7}$

$a_{1,6}$ $c_{7,8}$



Order in which the Vertices are popped of from the stack

f h g b e a d c

Reverse the order

Topological Sort

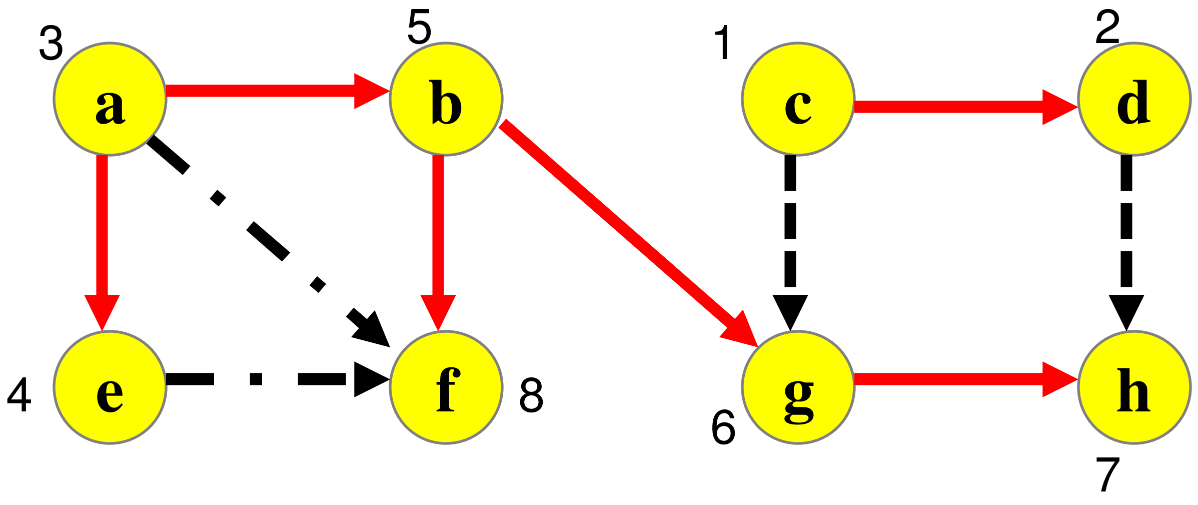
c d a e b g h f

DFS on a DAG

$$\begin{array}{c|cccccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\hline
1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
2 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
3 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
5 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
6 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}$$

Let the vertices be numbered in the order in which they are topologically sort.

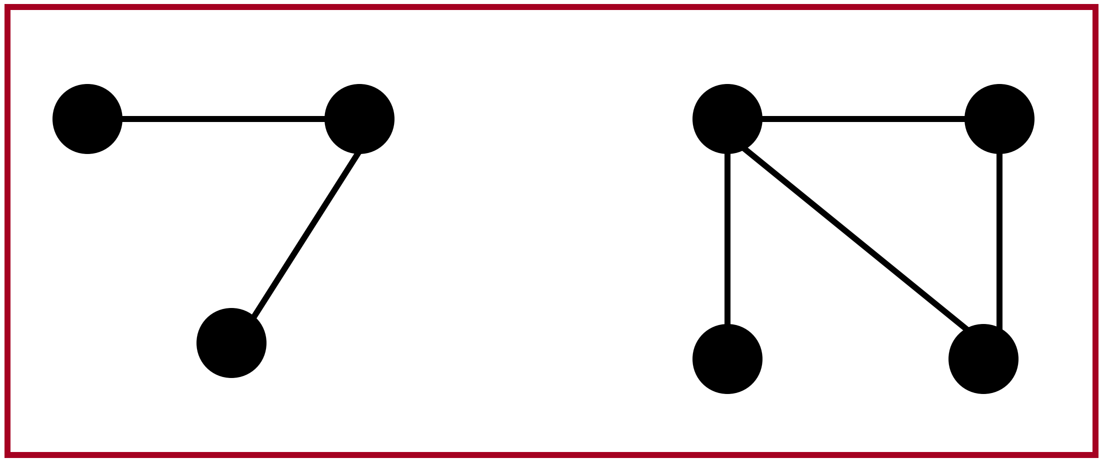
An adjacency matrix of the vertices listed in the order of their topological sort is strictly upper triangular.



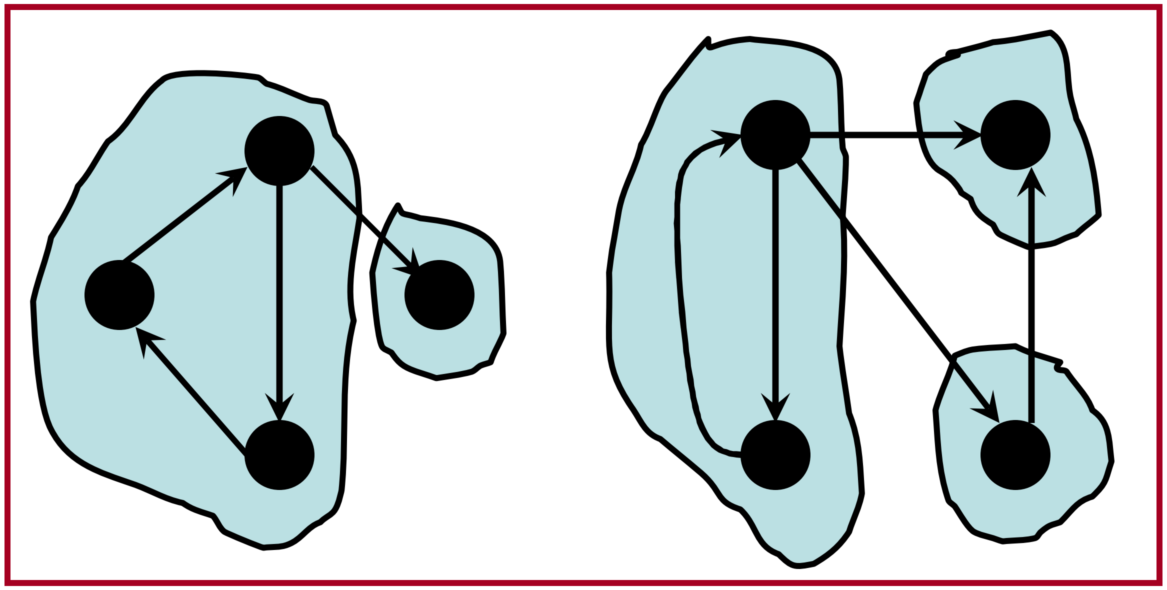
Components

- Component: The subset of vertices in a graph that are reachable from one another through paths of length one or more.
- Maximal subset property of a component: Inclusion of additional edges or vertices from the graph to this subset will break the connectivity of a component.
- A graph in which all its vertices are not in one component is said to be disconnected.
  - Undirected graph: The set of all vertices that are reachable (via BFS or DFS) starting from a particular vertex are all said to be in the same component.
  - Directed graph:
    - Weakly connected: The vertices of a di-graph are said to form a weakly connected component if they are connected in their undirected version.
    - Strongly connected: The vertices of a di-graph are said to form a strongly connected component if they are connected (reachable from each other through paths).
  - A standalone vertex that is not reachable to and from any other vertex is said to be in its own component.

Components: Examples



Undirected graph with Two components

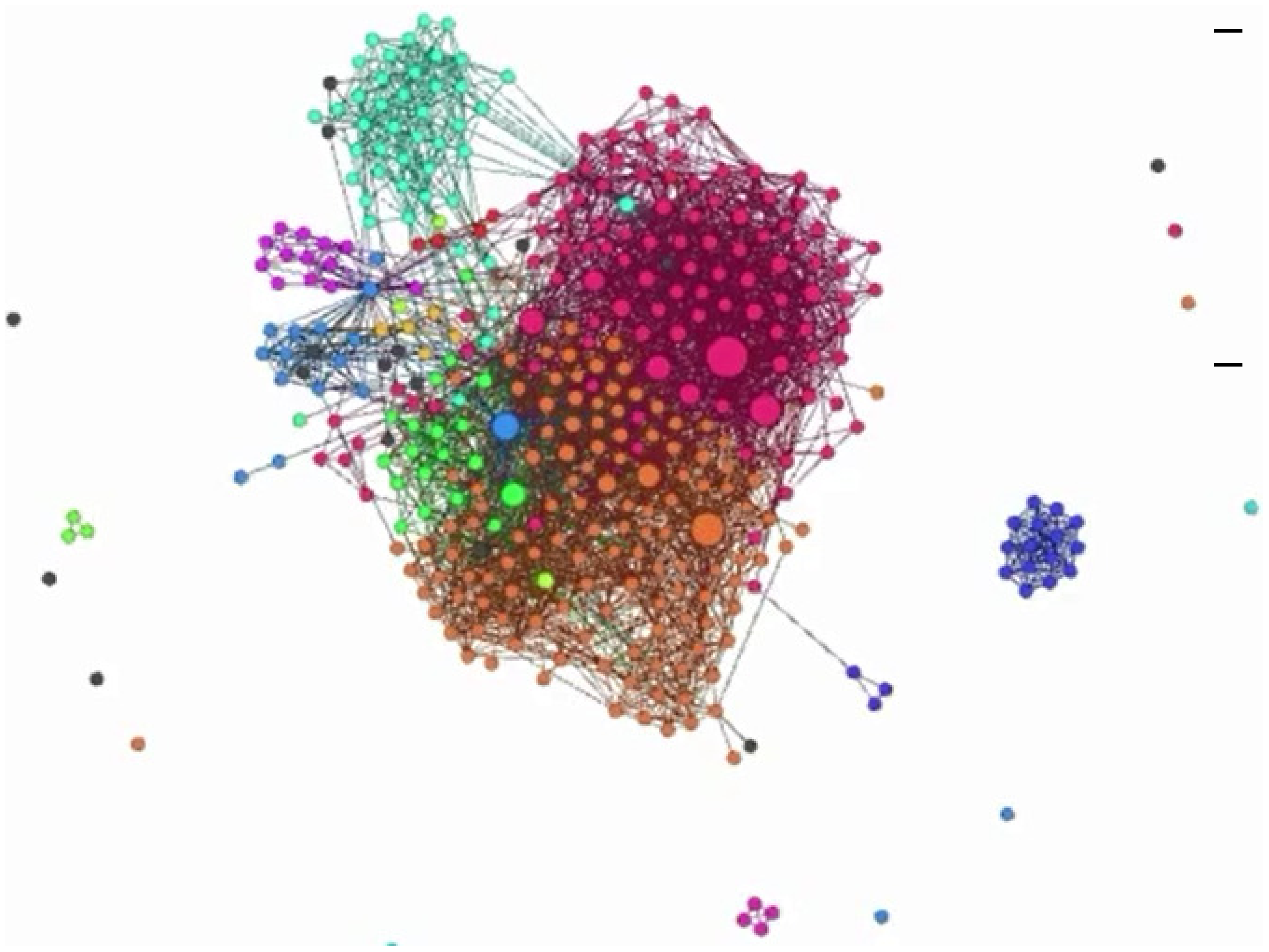


Directed graph with

Two weakly connected Components

Five strongly connected components

Connectivity: Good or bad?



- It depends:
  - Spread of good news – want to stay in a bigger connected component as in the center ones
  - Spread of virus – want to stay in a smaller connected component like the smaller ones so that you are less likely to be attacked.

Components in a DAG

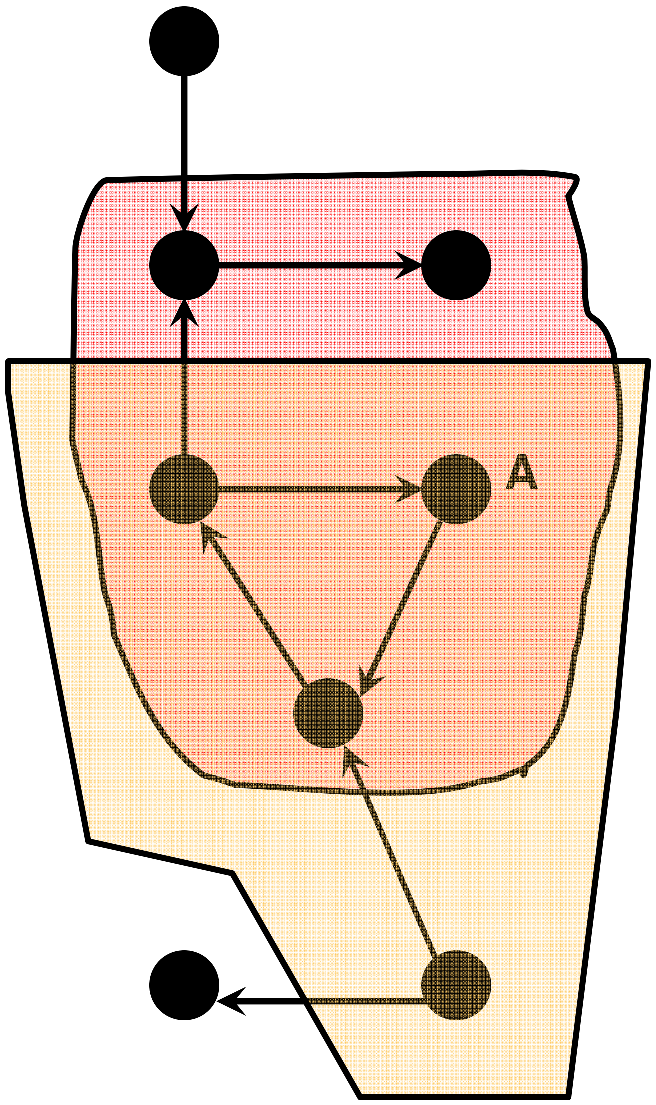
- Note that for a directed graph to have a strongly connected component (scc), the graph should have a cycle: because, we need the vertices in an scc to be reachable from one another.
  - There has to be a path from A to B through a sequence of edges and vice-versa through a different sequence of edges.
- Hence, there cannot be a strongly connected component in a DAG.

Out- and In- Components

- The out-component (defined for a specific vertex A) in a directed graph is the set of all vertices (including A itself) that are reachable from A through one or more paths.

- The in-component (defined for a specific vertex A) in a directed graph is the set of all vertices (including A itself) that have a directed path to A.

- The intersection of both the out-component and in-component of A yields a strongly connected component (scc) involving vertex A. All the vertices in such a scc are reachable from each other at least through A.



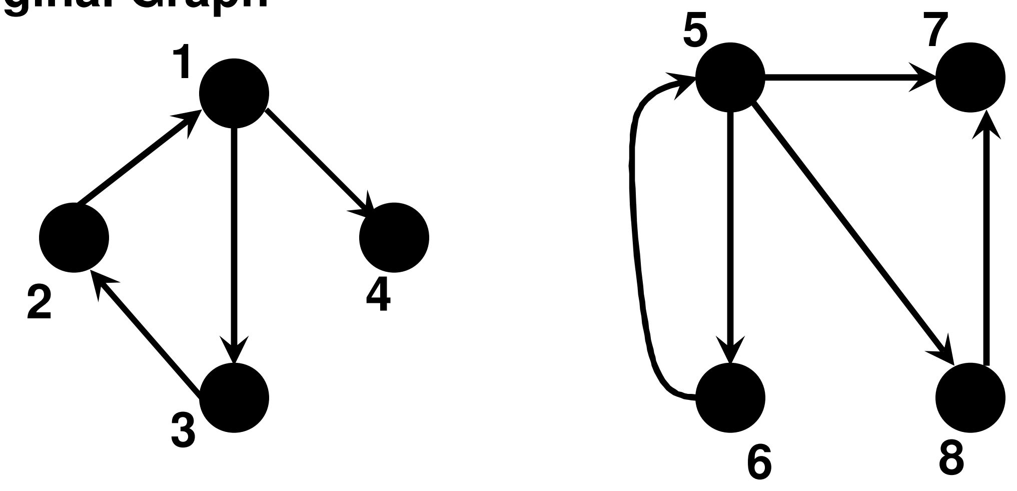
Algorithm to Find Strongly Connected Components in a Di-Graph

- Let G be a directed graph and SCC-Stack (S) be an empty stack.
- While SCC-Stack does not contain all vertices:
  - Choose an arbitrary vertex *v* not in S. Perform a DFS starting at *v*. Each time DFS pops out a vertex *u* from its stack (note: the DFS stack), push *u* onto the SCC-Stack.
- Reverse the directions of all edges to obtain the transpose graph of G.
- While the SCC-Stack is nonempty:
  - Pop the top vertex *v* from S. Perform a DFS starting at *v* in the transpose graph. The set of visited vertices will give the strongly connected component containing *v*; record this and remove all these vertices from the graph G and the stack S.

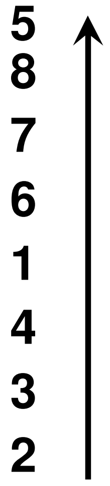
- Time complexity: Two DFS: $\Theta(V+E)$.

Finding Strongly Connected Components

Original Graph

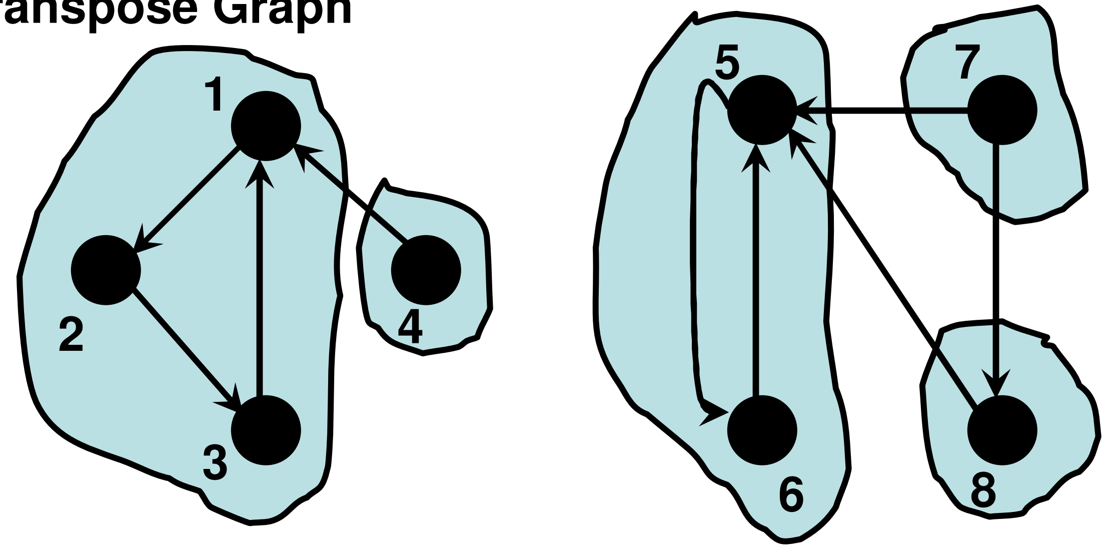


SCC Stack



Order in which the vertices are popped out of the DFS stack

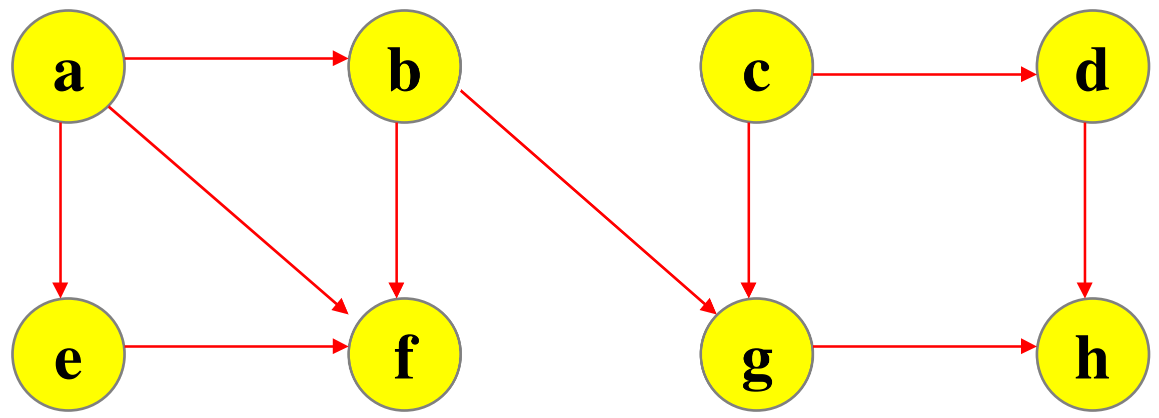
Transpose Graph



DFS Traversals

5 → 6
8
7
1 → 2 → 3
4

Finding Strongly Connected Components



$h_{5,2}$

$g_{4,3}$

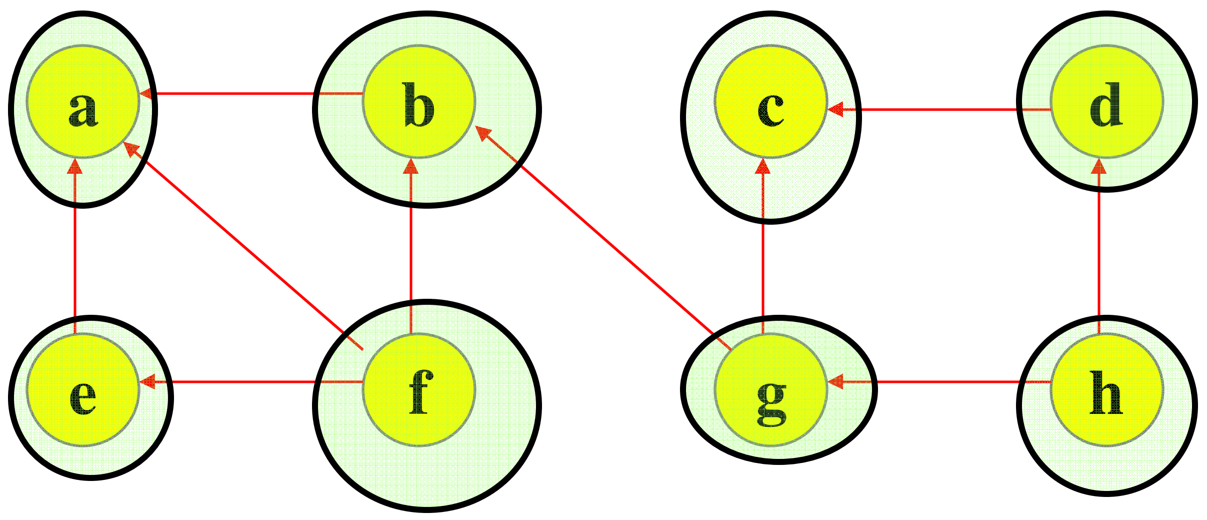
$f_{3,1}$

$b_{2,4}$ $e_{6,5}$ $d_{8,7}$

$a_{1,6}$ $c_{7,8}$

SCC Stack

c
d
a
e
b
g
h
f

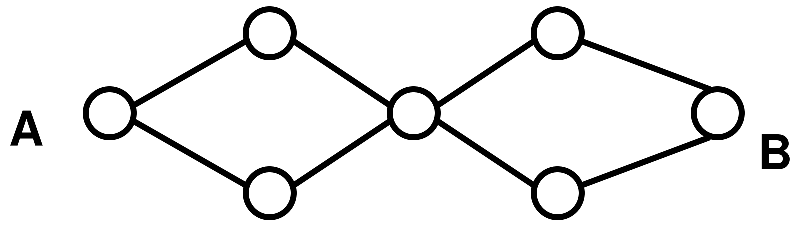


DFS Traversals

c
d
a
e
b
g
h
f

Vertex Connectivity, Edge Connectivity

- Consider the set of paths between two vertices A and B.
- Vertex-disjoint paths: Two paths are said to be vertex-disjoint if there is no common intermediate vertex between the two paths.
- Edge-disjoint paths: Two paths between are said to be edge-disjoint if there is no common edge between the two paths.
- Vertex Connectivity between two vertices A and B is the number of vertex-disjoint paths between A and B.
- Edge Connectivity between two vertices A and B is the number of edge-disjoint paths between A and B.



Vertex Connectivity – 1
Edge Connectivity – 2

Local Clustering Coefficient

- The local clustering coefficient captures the degree to which the neighbors of a given node link to each other.
- If k_i is the degree of node i, then the maximum number of links between its k_i neighbors is $k_i * (k_i – 1) / 2$.
- Let L_i be the number of links among the neighbors of node i. Local clustering coefficient of node i is

$$C_i = \frac{L_i}{\left[\frac{k_i(k_i - 1)}{2}\right]}$$

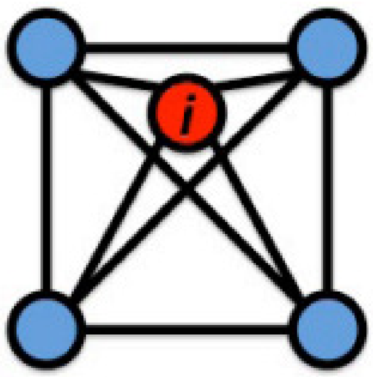
Local clustering coefficient is a measure of the neighborhood density. Larger the value, more dense is the neighborhood and vice-versa.

- The local clustering coefficient is a probability that any two neighbor nodes of a node are linked to each other.
- Average Clustering Coefficient

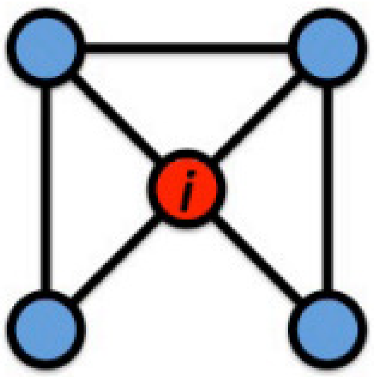
$$\langle C \rangle = \frac{1}{N}\sum_{i=1}^{N} C_i$$

Is a measure of the probability that any two neighbor nodes of a randomly selected node are linked to each other.

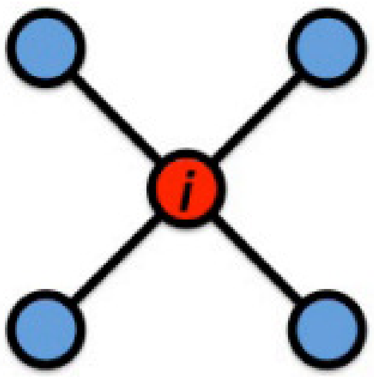
Examples for Local Clustering Coefficients



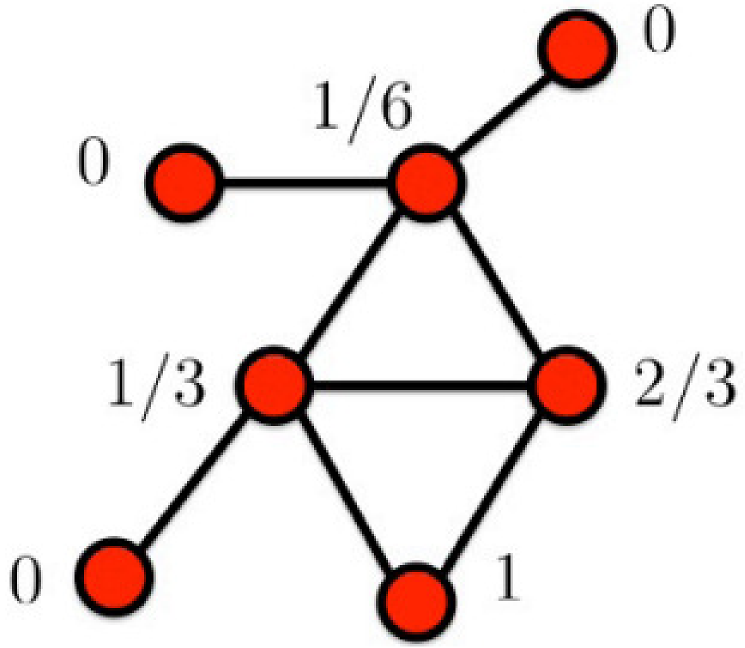
$C_i = 1$



$C_i = 1/2$

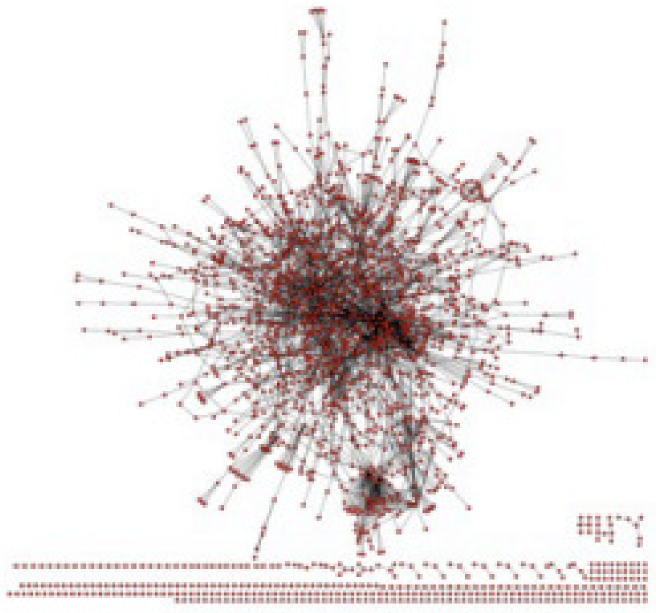


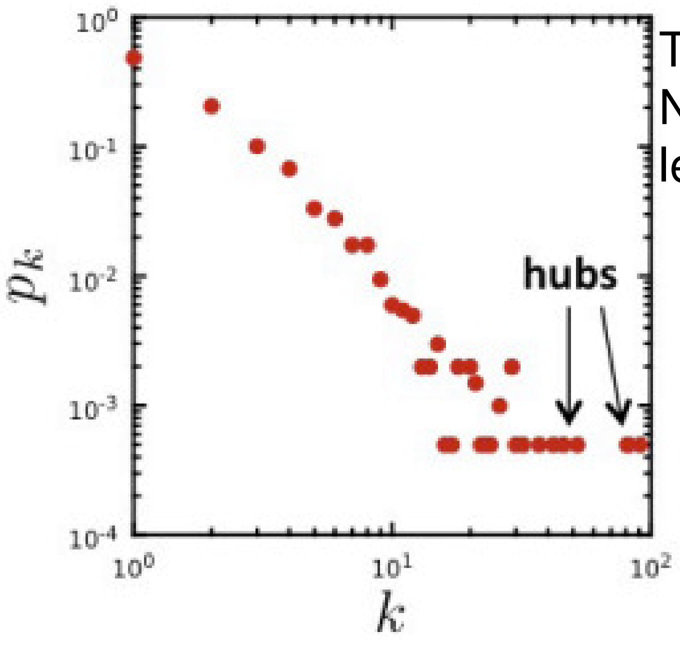
$C_i = 0$



$$\langle C \rangle = \frac{13}{42} \approx 0.310$$

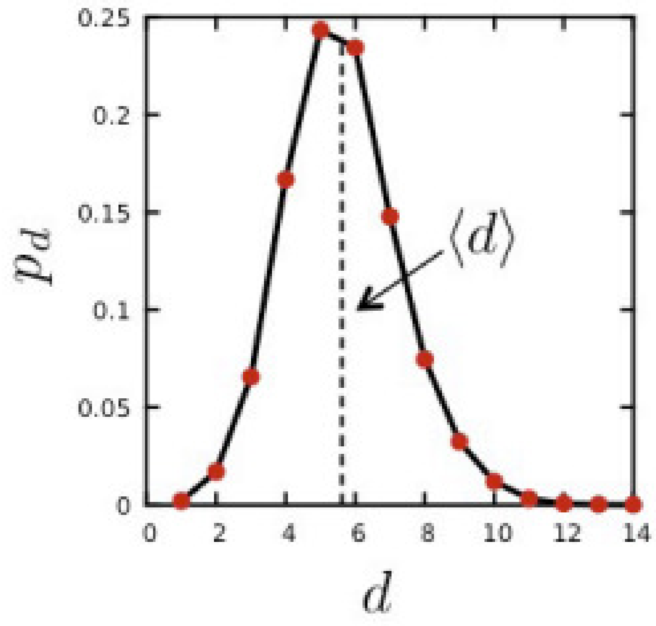
Case Study: PPI Network of Yeast

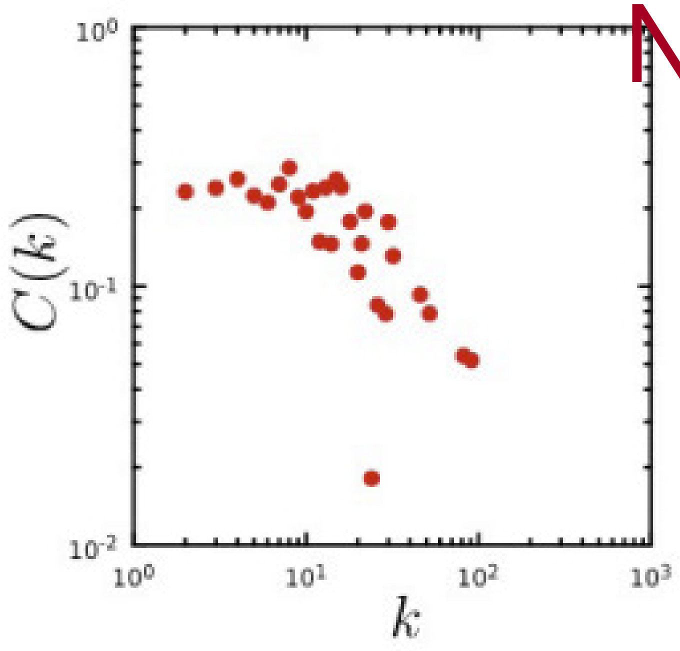




The probability of finding Nodes with degree less than 3 is 69%

Any two nodes are connected within a shorter distance





Nodes that have a high degree do not have a dense neighborhood. The contrary is observed.

Network Types: Terminologies

- Complement of a network G: Is a network comprising of all the nodes in G but comprising of links (between nodes) that are not in G.
- Regular network: An r-regular network is a network in which each node has degree r and such a network of n nodes has rn/2 links.
- Complete network: comprises of links between any two nodes
- Empty network: Complement of a complete network
- K-Colorability: A network is k-colorable if for any link in the network, the end vertices are colored in different colors, chosen among the k colors.
- Bipartite networks: The network is partitioned to two disjoint (non-empty) subsets V1 and V2 such that V1 U V2 = V, the set of all vertices and the only edges in the network are those that connect a vertex in V1 to a vertex in V2.
  - There are no edges between any two vertices within V1 or V2.
- K-partite networks: The network is partitioned into k-disjoint subsets; such that any edge in the network is between the vertices across any two of these subsets.