Social Network Analysis
Lecture 2-Introduction Graph Theory

Donglei Du
(ddu@unb.ca)

Faculty of Business Administration, University of New Brunswick, NB Canada Fredericton
E3B 9Y2

Table of contents

Layout

What is Graph theory?

What is Graph theory?

- Graph theory is the study of graphs, which are mathematical representation of a network used to model pairwise relations between objects.

What is Graph theory?

- Graph theory is the study of graphs, which are mathematical representation of a network used to model pairwise relations between objects.
- A *graph* consists of a set of "vertices" or "nodes", with certain pairs of these nodes connected by "edges" (undirected) or "arcs" (directed).

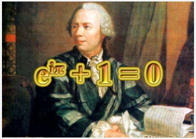
What is Graph theory?

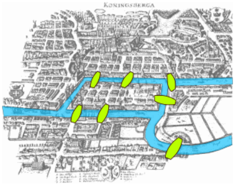
- Graph theory is the study of graphs, which are mathematical representation of a network used to model pairwise relations between objects.
- A *graph* consists of a set of "vertices" or "nodes", with certain pairs of these nodes connected by "edges" (undirected) or "arcs" (directed).
- A graph may be undirected, meaning that there is no distinction between the two vertices associated with each edge, or directed, meaning that its arcs may be directed from one vertex to another.

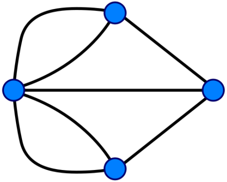
What is Graph theory?

- Graph theory is the study of graphs, which are mathematical representation of a network used to model pairwise relations between objects.
- A *graph* consists of a set of "vertices" or "nodes", with certain pairs of these nodes connected by "edges" (undirected) or "arcs" (directed).
- A graph may be undirected, meaning that there is no distinction between the two vertices associated with each edge, or directed, meaning that its arcs may be directed from one vertex to another.
- *This is an extremely brief introduction of graph theory*.

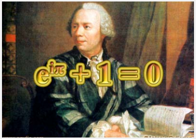
The Seven Bridges of Königsberg:

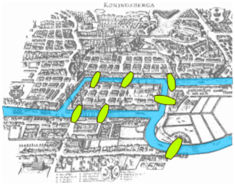


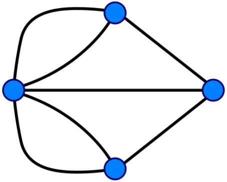




The Seven Bridges of Königsberg:

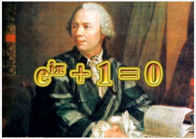


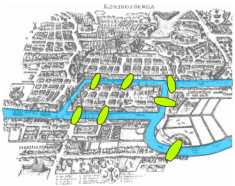


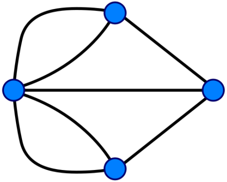


- **Q**: Is there a Eulerian trail through the city that would cross each bridge once and only once? (The second (undirected) graph represents the bridge network!)
    - **A**: an (connected) undirected graph has an Eulerian trail if and only if at most two vertices have odd degree.

The Seven Bridges of Königsberg:

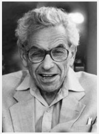


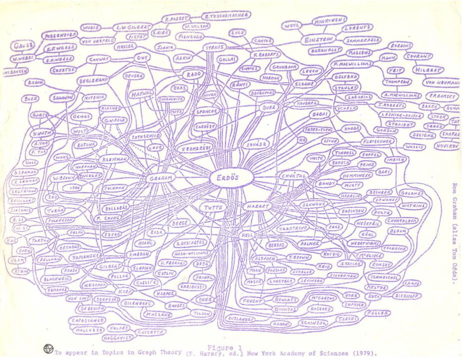




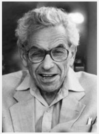
- **Q**: Is there a Eulerian trail through the city that would cross each bridge once and only once? (The second (undirected) graph represents the bridge network!)
  - **A**: an (connected) undirected graph has an Eulerian trail if and only if at most two vertices have odd degree.
- Leonhard Euler (15 April 1707-18 September 1783, Swiss Mathematician) in 1735 laid the foundations of graph theory and prefigured the idea of topology by studying this problem.

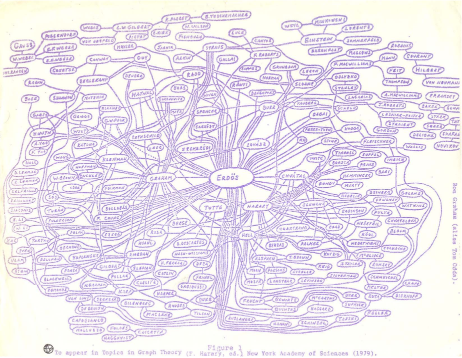
The Erdös coauthor network and Erdös number:





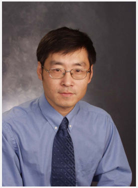
The Erdös coauthor network and Erdös number:



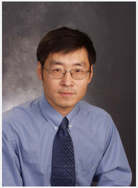


- Paul Erdös (26 March 191320 September 1996, Hungarian mathematician): one of the most prolific publishers of papers in mathematical history, comparable only with Leonhard Euler; Erdös published more papers, mostly in collaboration with other mathematicians, while Euler published more pages, mostly by himself.

My Erdös Number: 3

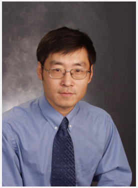


My Erdös Number: 3



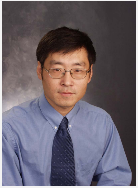
- Paul Erdös (**0**) and Prasad Tetali (**1**), Representations of integers as the sum of k terms, *Random Structures Algorithms*, 1(3) (1990), 245-261.

My Erdös Number: 3



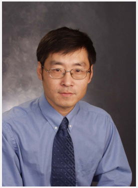
- Paul Erdös (**0**) and Prasad Tetali (**1**), Representations of integers as the sum of k terms, *Random Structures Algorithms*, 1(3) (1990), 245-261.
- Prasad Tetali (**1**), Juan C. Vera (**2**), Eric Vigoda, and Linji Yang, Phase transition for the mixing time of the Glauber dynamics for coloring regular trees, *Annals of Applied Probability*, 22(6) (2012), 2210-2239.

My Erdös Number: 3



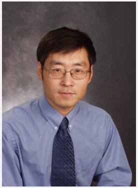
- Paul Erdös (**0**) and Prasad Tetali (**1**), Representations of integers as the sum of k terms, *Random Structures Algorithms*, 1(3) (1990), 245-261.
- Prasad Tetali (**1**), Juan C. Vera (**2**), Eric Vigoda, and Linji Yang, Phase transition for the mixing time of the Glauber dynamics for coloring regular trees, *Annals of Applied Probability*, 22(6) (2012), 2210-2239.
- Qiaming Han, Donglei Du (**3**), Juan C. Vera (**2**) and Luis F. Zuluag, Improved bounds for the symmetric rendezvous search problem on the line, *Operations Research*, 56(3) (2008), 772-782.

My Erdös Number: 3



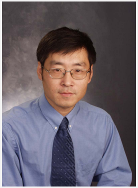
- Paul Erdös (**0**) and Prasad Tetali (**1**), Representations of integers as the sum of k terms, *Random Structures Algorithms*, 1(3) (1990), 245-261.
- Prasad Tetali (**1**), Juan C. Vera (**2**), Eric Vigoda, and Linji Yang, Phase transition for the mixing time of the Glauber dynamics for coloring regular trees, *Annals of Applied Probability*, 22(6) (2012), 2210-2239.
- Qiaming Han, Donglei Du (**3**), Juan C. Vera (**2**) and Luis F. Zuluag, Improved bounds for the symmetric rendezvous search problem on the line, *Operations Research*, 56(3) (2008), 772-782.
- Find your own Erdös Number

My Erdös Number: 3



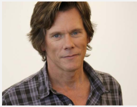
- Paul Erdös (**0**) and Prasad Tetali (**1**), Representations of integers as the sum of k terms, *Random Structures Algorithms*, 1(3) (1990), 245-261.
- Prasad Tetali (**1**), Juan C. Vera (**2**), Eric Vigoda, and Linji Yang, Phase transition for the mixing time of the Glauber dynamics for coloring regular trees, *Annals of Applied Probability*, 22(6) (2012), 2210-2239.
- Qiaming Han, Donglei Du (**3**), Juan C. Vera (**2**) and Luis F. Zuluag, Improved bounds for the symmetric rendezvous search problem on the line, *Operations Research*, 56(3) (2008), 772-782.
- Find your own Erdös Number
  - here: http://www.ams.org/mathscinet/collaborationDistance.html

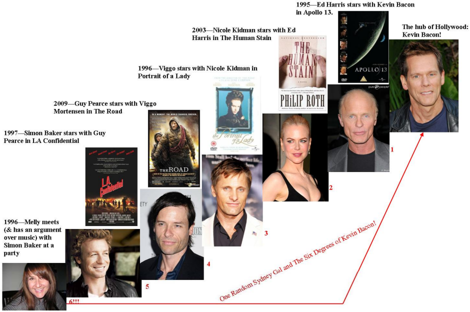
My Erdös Number: 3



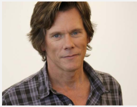
- Paul Erdös (**0**) and Prasad Tetali (**1**), Representations of integers as the sum of k terms, *Random Structures Algorithms*, 1(3) (1990), 245-261.
- Prasad Tetali (**1**), Juan C. Vera (**2**), Eric Vigoda, and Linji Yang, Phase transition for the mixing time of the Glauber dynamics for coloring regular trees, *Annals of Applied Probability*, 22(6) (2012), 2210-2239.
- Qiaming Han, Donglei Du (**3**), Juan C. Vera (**2**) and Luis F. Zuluag, Improved bounds for the symmetric rendezvous search problem on the line, *Operations Research*, 56(3) (2008), 772-782.
- Find your own Erdös Number
  - here: http://www.ams.org/mathscinet/collaborationDistance.html
- If you want to know more about Erdös Number,try here: http://www.oakland.edu/enp/compute/

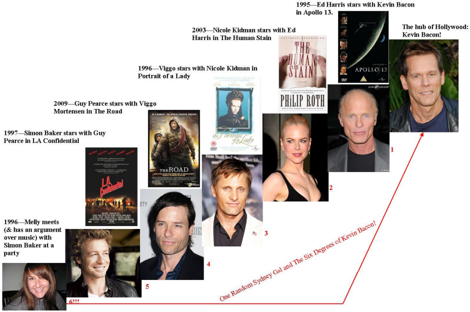
Six Degrees of Kevin Bacon:





Six Degrees of Kevin Bacon:



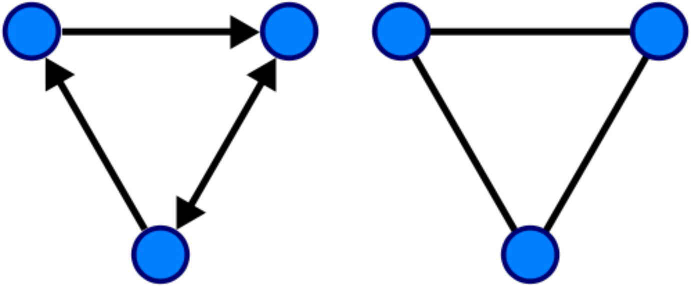


- Kevin Norwood Bacon (July 8, 1958-) is an American actor and musician

Types of graphs

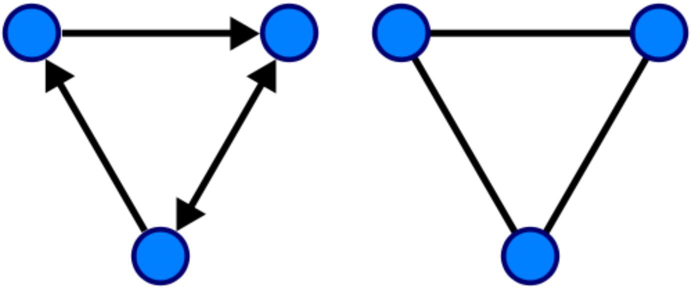
Types of graphs

- Undirected vs directed:



Types of graphs

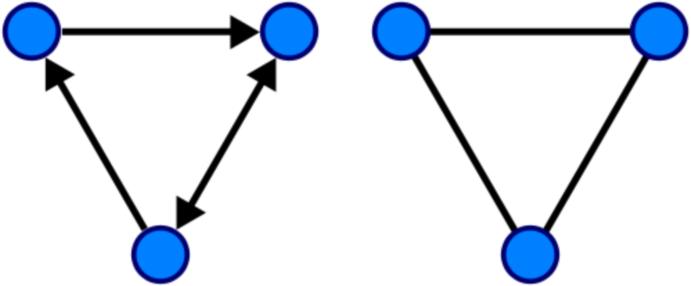
- Undirected vs directed:



- Undirected networks: coauthorship network, actor network, Königsberg Bridges Network, Facebook friendship network

Types of graphs

- Undirected vs directed:

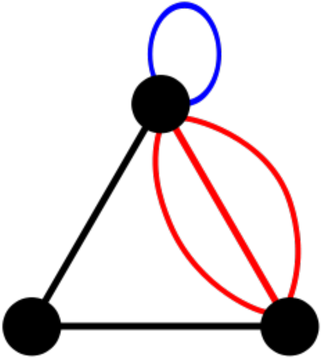


- Undirected networks: coauthorship network, actor network, Königsberg Bridges Network, Facebook friendship network
- Directed networks: URLs on the www, phone calls, Retweet network

Types of graphs

Types of graphs

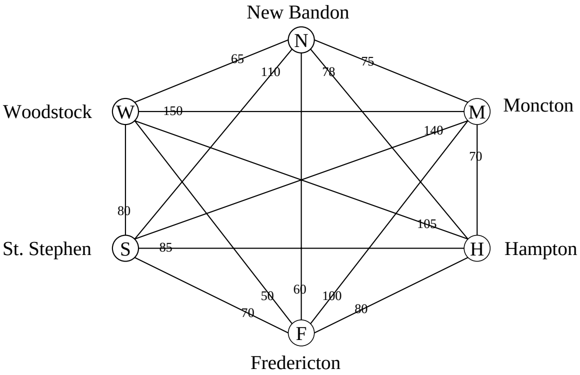
- Simple vs multigraph: lops or multiedges



Types of graphs

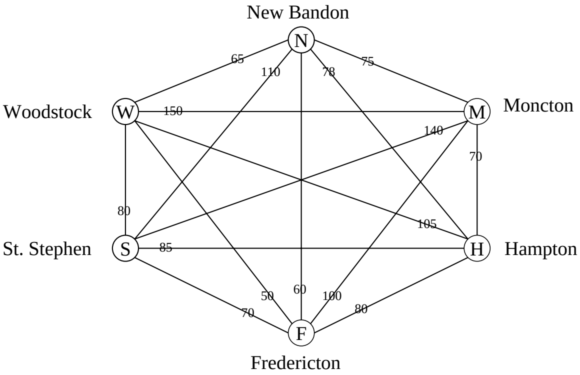
Types of graphs

- Unweighted vs weighted:



Types of graphs

- Unweighted vs weighted:



- Mobile phone calls, Collaboration network

Important graphs

Important graphs

- Regular graph

Important graphs

- Regular graph
- Complete graph

Important graphs

- Regular graph
- Complete graph
- Path

Important graphs

- Regular graph
- Complete graph
- Path
- Cycle

Important graphs

- Regular graph
- Complete graph
- Path
- Cycle
- Bipartitie graph

Important graphs

- Regular graph
- Complete graph
- Path
- Cycle
- Bipartitie graph
- Euler graph

Important graphs

- Regular graph
- Complete graph
- Path
- Cycle
- Bipartitie graph
- Euler graph
- Hamilton graph

Important graphs

- Regular graph
- Complete graph
- Path
- Cycle
- Bipartitie graph
- Euler graph
- Hamilton graph
- Planar graph

Important graphs

- Regular graph
- Complete graph
- Path
- Cycle
- Bipartitie graph
- Euler graph
- Hamilton graph
- Planar graph
- Tree and forest

Important graphs

- Regular graph
- Complete graph
- Path
- Cycle
- Bipartitie graph
- Euler graph
- Hamilton graph
- Planar graph
- Tree and forest
- ...

Layout

Representation of graphs using different data structure (Hi, Computer Science guys)

Representation of graphs using different data structure (Hi, Computer Science guys)

- Graph can be represented in many different ways for different purpose

Representation of graphs using different data structure (Hi, Computer Science guys)

- Graph can be represented in many different ways for different purpose
  - Adjacency matrix

Representation of graphs using different data structure (Hi, Computer Science guys)

- Graph can be represented in many different ways for different purpose
  - Adjacency matrix
  - Edge list

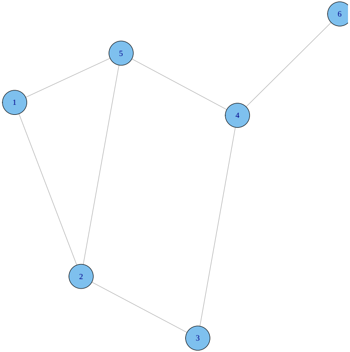
Representation of graphs using different data structure (Hi, Computer Science guys)

- Graph can be represented in many different ways for different purpose
  - Adjacency matrix
  - Edge list
  - Adjacency list

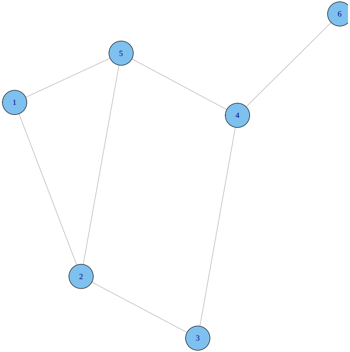
Representation of graphs using different data structure (Hi, Computer Science guys)

- Graph can be represented in many different ways for different purpose
  - Adjacency matrix
  - Edge list
  - Adjacency list
  - Laplace matrix

Adjacency matrix: undirected graph



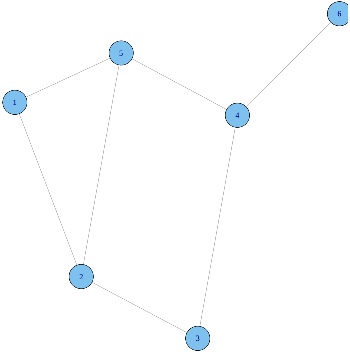
Adjacency matrix: undirected graph



- Adjacency matrix: $A_{ij} = 1$ iff there is a link between i and j.

$$
\begin{array}{c|c}
 & \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} &
\begin{bmatrix}
0 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0
\end{bmatrix}
\end{array}
$$

Adjacency matrix: undirected graph



- Adjacency matrix: $A_{ij} = 1$ iff there is a link between i and j.

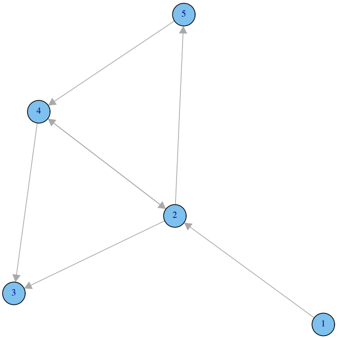
$$
\begin{array}{c}
\begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\
\begin{array}{c} 1\\2\\3\\4\\5\\6 \end{array}
\begin{bmatrix}
0 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0
\end{bmatrix}
\end{array}
$$

- The adjacency is symmetric for undirected graph.

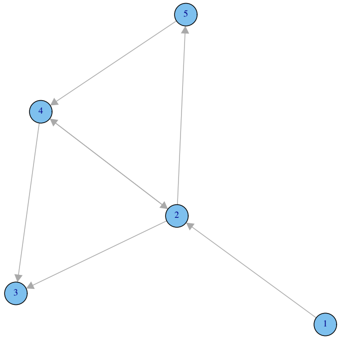
R code based on package igraph: generate graph from adjacency matrix

```
library(igraph)
#Generate graph object from adjacency matrix
adjm_u<-matrix(
   c(0, 1, 0, 0, 1, 0,
1, 0, 1, 0, 1, 0,
0, 1, 0, 1, 0, 0,
0, 0, 1, 0, 1, 1,
1, 1, 0, 1, 0, 0,
0, 0, 0, 1, 0, 0), # the data elements
   nrow=6,              # number of rows
   ncol=6,              # number of columns
   byrow = TRUE)        # fill matrix by rows
g_adj_u <- graph.adjacency(adjm_u, mode="undirected")
tkplot(g_adj_u)
```

Adjacency matrix: directed graph



Adjacency matrix: directed graph



- Adjacency matrix: $A_{ij} = 1$ iff there is a link from j to i

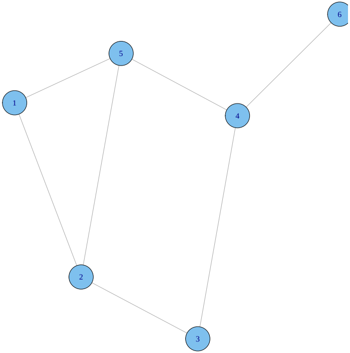
$$\begin{array}{c|c} & \begin{matrix} 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

- Note the direction of the edge runs from the second index to the first: counter-intuitive, but convenient mathematically!

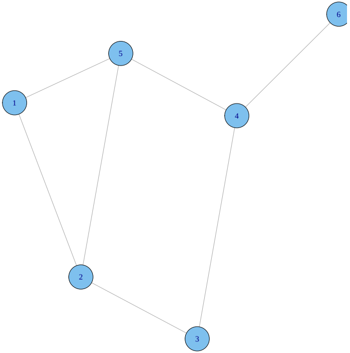
R code based on package igraph: generate graph from adjacency matrix

```
library(igraph)
#Generate graph object from adjacency matrix
adjm_d<-matrix(
   c(0, 1, 0, 0, 0,
0, 0, 1, 1, 1,
0, 0, 0, 0, 0,
0, 1, 1, 0, 0,
0, 0, 0, 1, 0), # the data elements
   nrow=5,              # number of rows
   ncol=5,              # number of columns
   byrow = TRUE)        # fill matrix by rows
g_adj_d <- graph.adjacency(adjm_d, mode="directed")
tkplot(g_adj_d)
```

Edge list: undirected graph



Edge list: undirected graph



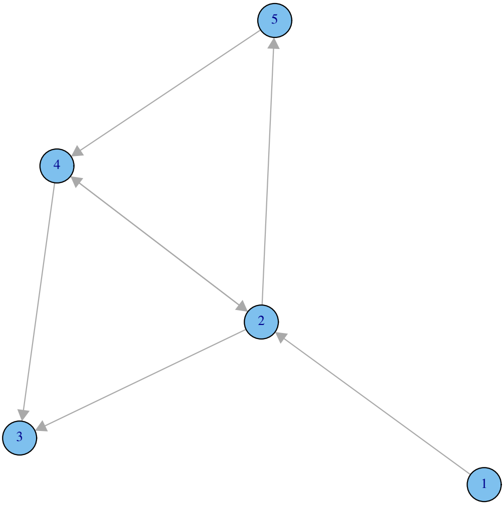
- Edge list:

(1,2)
(1,5)
(2,3)
(2,5)
(3,4)
(4,5)
(4,6)

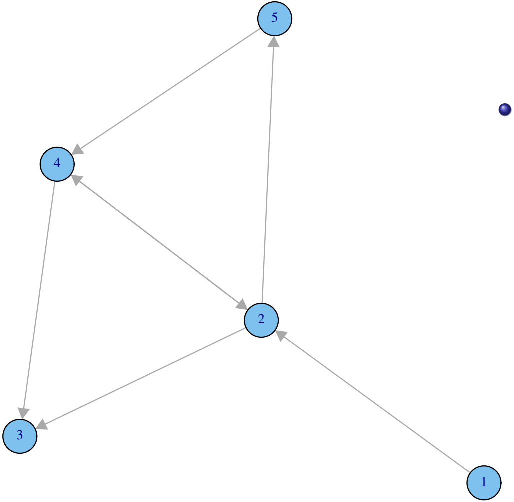
R code based on package igraph: generate undirected graph from edge list

```
library(igraph)
#Generate graph object from edge list
el_u <- matrix( c(1, 2, 1, 5, 2,3,2,5,3,4,4,5,4,6), nc=2,
g_el_u <- graph.edgelist(el_u,directed=FALSE)
tkplot(g_el_u)
```

Edge list: directed graph



Edge list: directed graph



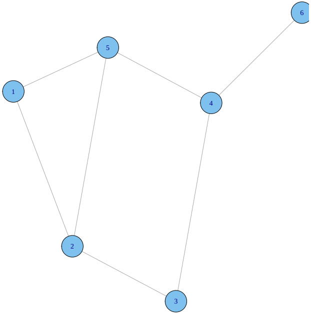
- Edge list:

  12
  23
  24
  42
  25
  43
  54

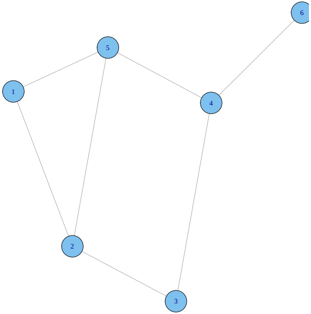
R code based on package igraph: generate directed graph from edge list

```
library(igraph)
#Generate graph object from edge list
el_d <- matrix( c(1, 2, 2, 3, 2,4,4,2,2,5,4,3,5,4), nc=2,
g_el_d <- graph.edgelist(el_d,directed=TRUE)
tkplot(g_el_d)
```

Adjacency list: undirected graph



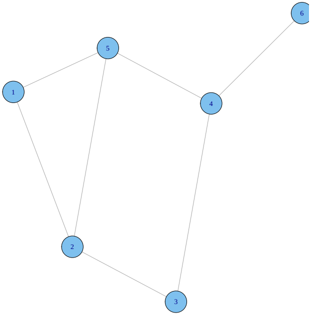
Adjacency list: undirected graph



- Adjacency list:

node	neighbor
1:	2 5
2:	1 3 5
3:	2 4
4:	3 5 6
5:	1 2 4
6:	4

Adjacency list: undirected graph

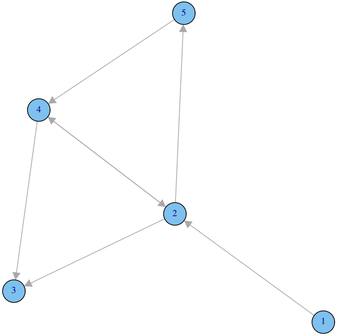


- Adjacency list:

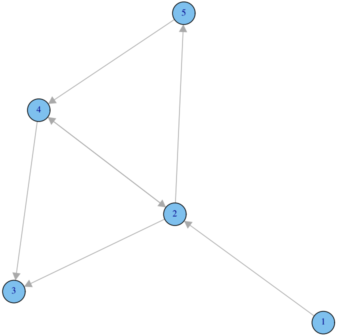
node	neighbor
1:	2 5
2:	1 3 5
3:	2 4
4:	3 5 6
5:	1 2 4
6:	4

- Easier to work with if network is large and parse, and quick in retrieving all neighbors for a node

Adjacency list: directed graph



Adjacency list: directed graph



- Adjacency list:

node	inneighbor	outneighbor
1:		2
2:	1,4	3,4,5
3:	2,4	
4:	2,5	2,3
5:	2	4

Layout

Graph-theoretic concepts

Graph-theoretic concepts

- Density

Graph-theoretic concepts

- Density
- Degree, indegree and outdegree

Graph-theoretic concepts

- Density
- Degree, indegree and outdegree
- Path and cycles

Graph-theoretic concepts

- Density
- Degree, indegree and outdegree
- Path and cycles
- Distance, diameter

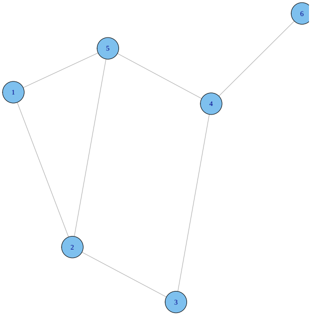
Graph-theoretic concepts

- Density
- Degree, indegree and outdegree
- Path and cycles
- Distance, diameter
- Components

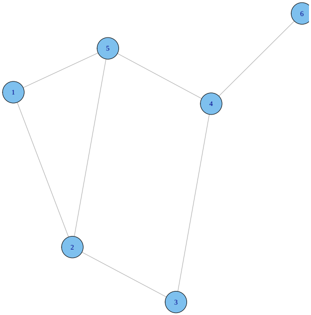
Graph-theoretic concepts

- Density
- Degree, indegree and outdegree
- Path and cycles
- Distance, diameter
- Components
- Clustering coefficient

Degree for undirected graph

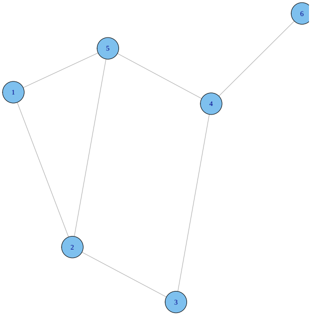


Degree for undirected graph



- Degree of any node i: the number of nodes adjacent to i. It can be calculated from the three different representations discussed earlier.

Degree for undirected graph



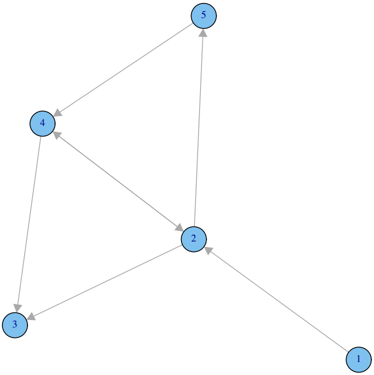
- Degree of any node i: the number of nodes adjacent to i. It can be calculated from the three different representations discussed earlier.
  - Every loop adds two degrees to a node.

node	degree
1	2
2	3
3	2
4	3
5	3
6	1

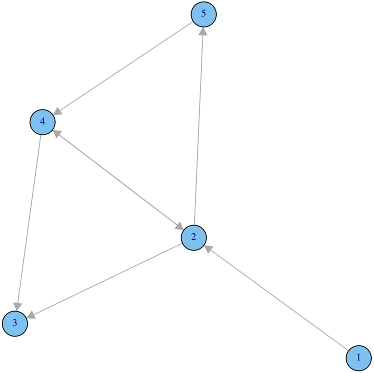
R code based on package igraph: degree

```
degree(g_adj_u)
```

Indegree and outdegree for directed graph

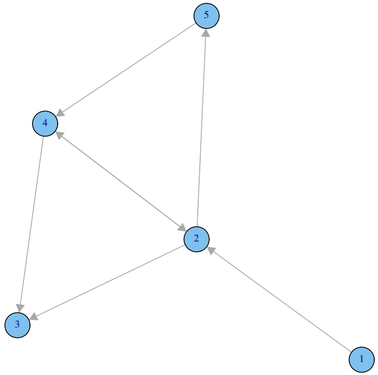


Indegree and outdegree for directed graph



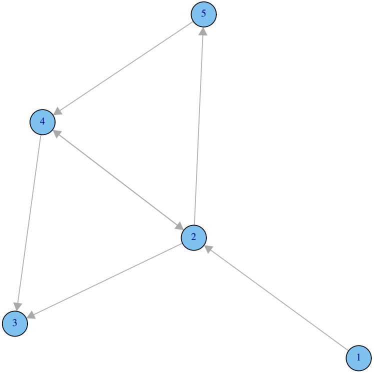
- Indegree of any node i: the number of nodes destined to i.

Indegree and outdegree for directed graph



- Indegree of any node i: the number of nodes destined to i.
- Outdegree of any node i: the number of nodes originated at i.

Indegree and outdegree for directed graph



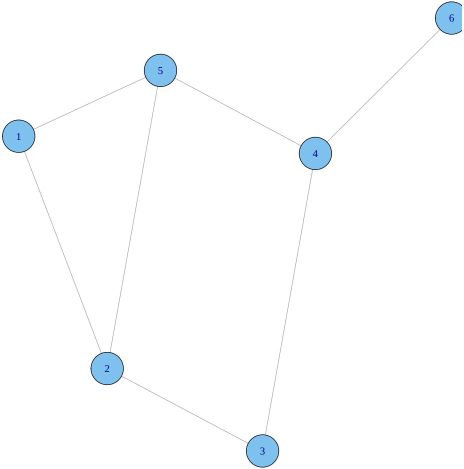
- Indegree of any node i: the number of nodes destined to i.
- Outdegree of any node i: the number of nodes originated at i.
  - Every loop adds one degree to each of the indegree and outdegree of a node.

node	indegree	outdegree
1	0	1
2	2	3
3	2	0
4	2	2
5	1	1

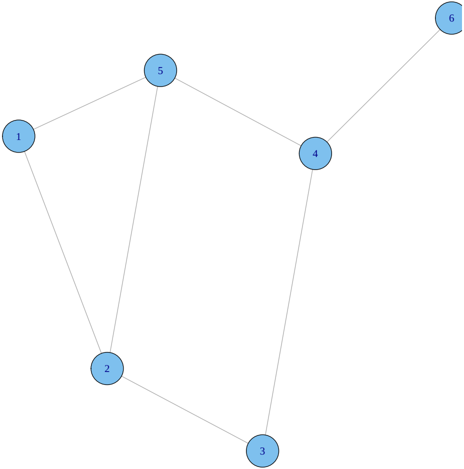
R code based on package igraph: degree

```
degree(g_adj_d, mode="in")
degree(g_adj_d, mode="out")
```

Walk and Path

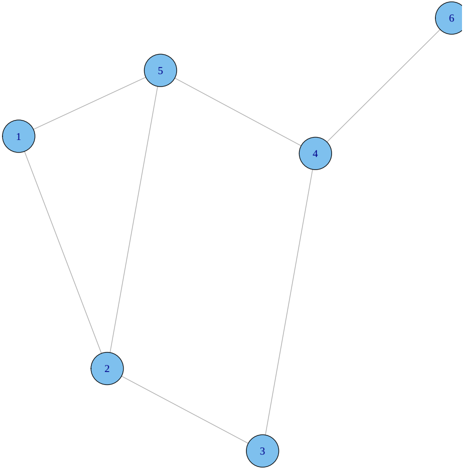


Walk and Path



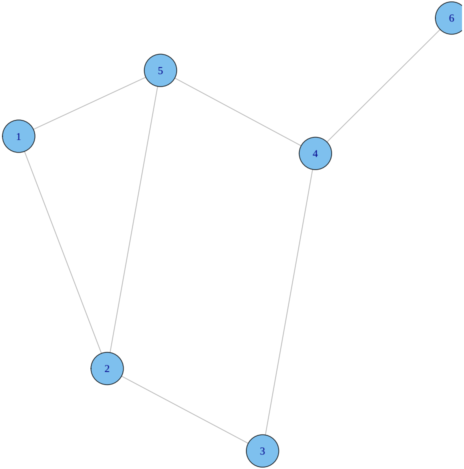
- Walk: a walk is a sequence of nodes in which each node is adjacent to the next one

Walk and Path



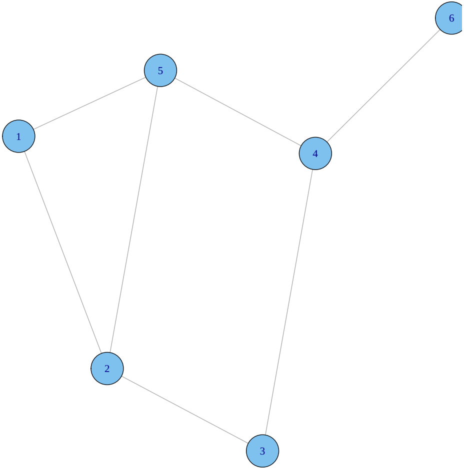
- Walk: a walk is a sequence of nodes in which each node is adjacent to the next one
  - A walk can intersect itself and pass through the same link repeatedly. Each time a link is crossed, it is counted separately

Walk and Path



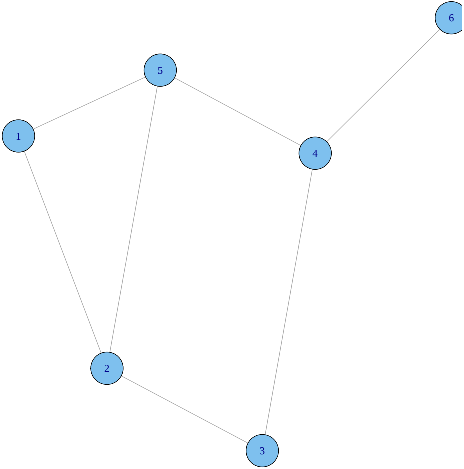
- Walk: a walk is a sequence of nodes in which each node is adjacent to the next one
  - A walk can intersect itself and pass through the same link repeatedly. Each time a link is crossed, it is counted separately
  - One walk on the graph: $1, 2, 5, 1, 2, 3, 4$.

Walk and Path



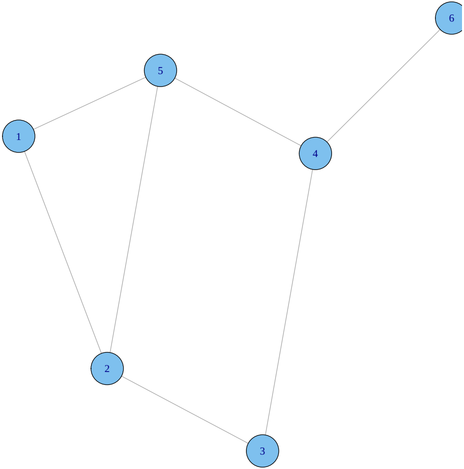
- Walk: a walk is a sequence of nodes in which each node is adjacent to the next one
  - A walk can intersect itself and pass through the same link repeatedly. Each time a link is crossed, it is counted separately
  - One walk on the graph: $1, 2, 5, 1, 2, 3, 4$.
  - In a directed network, the walk can follow only the direction of an arrow.

Walk and Path



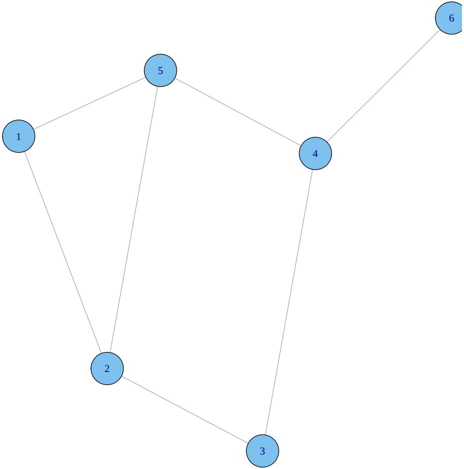
- Walk: a walk is a sequence of nodes in which each node is adjacent to the next one
  - A walk can intersect itself and pass through the same link repeatedly. Each time a link is crossed, it is counted separately
  - One walk on the graph: $1, 2, 5, 1, 2, 3, 4$.
  - In a directed network, the walk can follow only the direction of an arrow.
- A path is a walk without passing through the same link more than once (e.g. $1, 2, 5, 4, 3.$).

Walk and Path

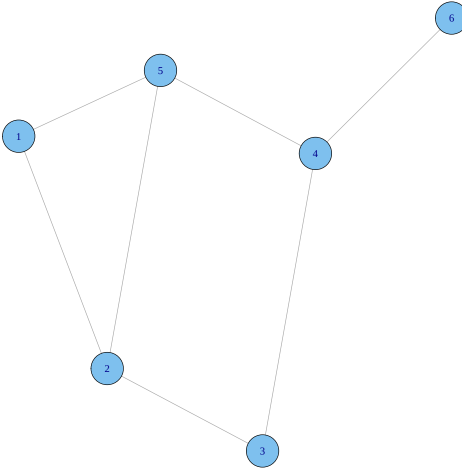


- Walk: a walk is a sequence of nodes in which each node is adjacent to the next one
  - A walk can intersect itself and pass through the same link repeatedly. Each time a link is crossed, it is counted separately
  - One walk on the graph: $1, 2, 5, 1, 2, 3, 4$.
  - In a directed network, the walk can follow only the direction of an arrow.
- A path is a walk without passing through the same link more than once (e.g. $1, 2, 5, 4, 3.$).
- Cycle: A path with the same start and end node (e.g., $1, 2, 5$)

Distance

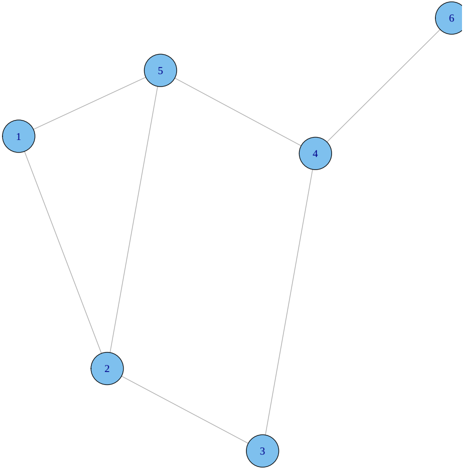


Distance



- The distance d_{ij} (shortest path, geodesic path) between two nodes i and j is the number of edges along the shortest path connecting them.

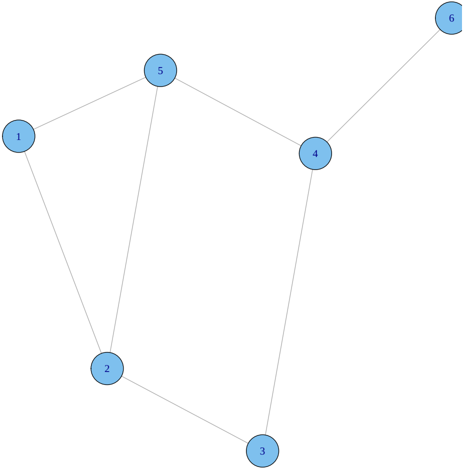
Distance



- The distance d_{ij} (shortest path, geodesic path) between two nodes i and j is the number of edges along the shortest path connecting them.
  - The Distance between every pair of nodes are

	1	2	3	4	5	6
1	0	1	2	2	1	3
2	1	0	1	2	1	3
3	2	1	0	1	2	2
4	2	2	1	0	1	1
5	1	1	2	1	0	2
6	3	3	2	1	2	0

Distance



- The distance d_{ij} (shortest path, geodesic path) between two nodes i and j is the number of edges along the shortest path connecting them.
  - The Distance between every pair of nodes are

	1	2	3	4	5	6
1	0	1	2	2	1	3
2	1	0	1	2	1	3
3	2	1	0	1	2	2
4	2	2	1	0	1	1
5	1	1	2	1	0	2
6	3	3	2	1	2	0

- For directed graphs, the distance from one node A to another B is generally different from that from B to A.

R code based on package igraph: distance

```
shortest.paths(g_adj_u)
```

Number of walks of length k

Number of walks of length k

- If A is the adjacency matrix of the directed or undirected graph G, then the matrix A^k (i.e., the matrix product of k copies of A) has an interesting interpretation:

Number of walks of length k

- If A is the adjacency matrix of the directed or undirected graph G, then the matrix A^k (i.e., the matrix product of k copies of A) has an interesting interpretation:
- the entry in row i and column j gives the number of (directed or undirected) walks of length k from vertex i to vertex j.

Number of walks of length k

- If A is the adjacency matrix of the directed or undirected graph G, then the matrix A^k (i.e., the matrix product of k copies of A) has an interesting interpretation:
- the entry in row i and column j gives the number of (directed or undirected) walks of length k from vertex i to vertex j.
- This implies, for example, that the number of triangles in an undirected graph G is exactly the trace of $A^3/3!$.

Connectivity and connected components for undirected graph

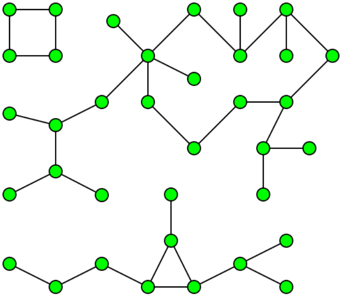


Figure: A graph with three connected components

Connectivity and connected components for undirected graph

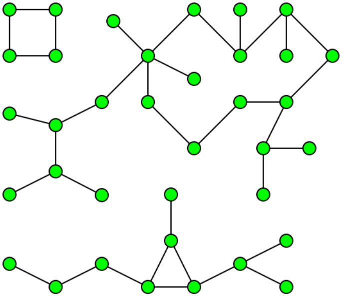


Figure: A graph with three connected components

- A graph is connected if there is a path between every pair of nodes.

Connectivity and connected components for undirected graph

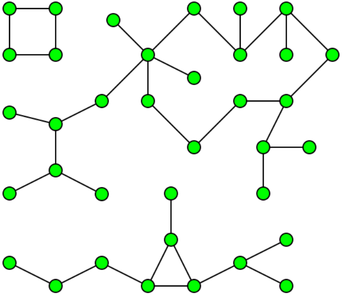


Figure: A graph with three connected components

- A graph is connected if there is a path between every pair of nodes.
- A connected component is a subgraph in which any two nodes are connected to each other by paths, and which is connected to no additional nodes in the original graph.

Connectivity and connected components for undirected graph

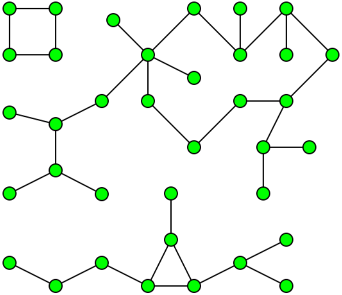


Figure: A graph with three connected components

- A graph is connected if there is a path between every pair of nodes.
- A connected component is a subgraph in which any two nodes are connected to each other by paths, and which is connected to no additional nodes in the original graph.
- Largest Component: Giant Component

Connectivity and connected components for undirected graph

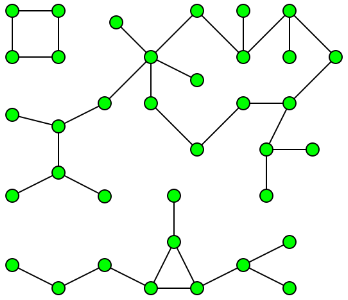


Figure: A graph with three connected components

- A graph is connected if there is a path between every pair of nodes.
- A connected component is a subgraph in which any two nodes are connected to each other by paths, and which is connected to no additional nodes in the original graph.
- Largest Component: Giant Component
- Bridge: an edge whose deletion increases the number of connected components.

Connectivity and connected components for directed graph

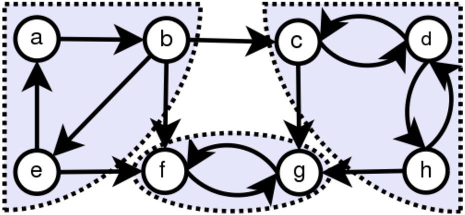


Figure: A graph with three strongly connected components

Connectivity and connected components for directed graph

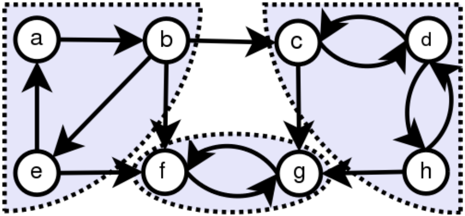


Figure: A graph with three strongly connected components

- A directed graph is strongly connected if there is a path from any node to each other node, and vice versa.

Connectivity and connected components for directed graph

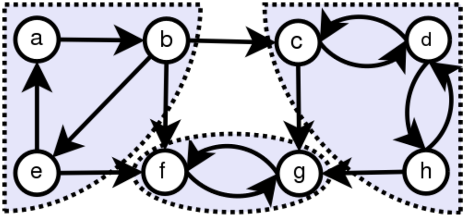


Figure: A graph with three strongly connected components

- A directed graph is strongly connected if there is a path from any node to each other node, and vice versa.
- A directed graph is weakly connected if it is connected by ignoring the edge directions

Connectivity and connected components for directed graph

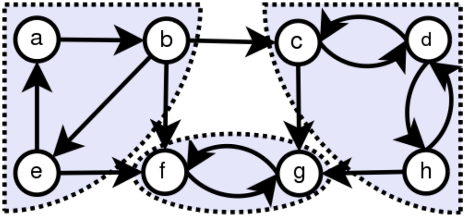
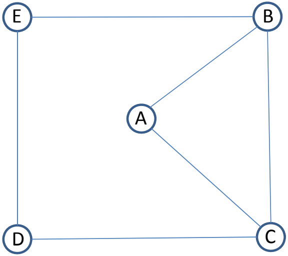


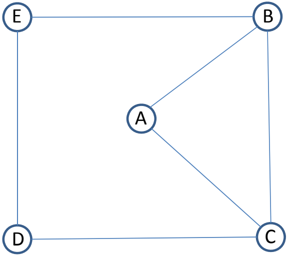
Figure: A graph with three strongly connected components

- A directed graph is strongly connected if there is a path from any node to each other node, and vice versa.
- A directed graph is weakly connected if it is connected by ignoring the edge directions
- A strongly connected component of a directed graph G is a subgraph that is strongly connected, and is maximal with this property: no additional edges or vertices from G can be included in the subgraph without breaking its property of being strongly directed.

Clustering coefficient



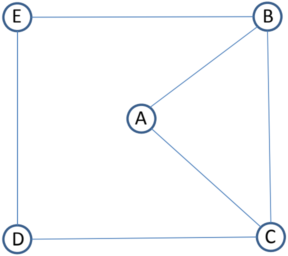
Clustering coefficient



- For any given node A and two randomly selected nodes B and C:

$$
\begin{aligned}
CC(A) &= \mathbb{P}(B \in N(C) | B, C \in N(A)) \\
&= \mathbb{P}(\text{two randomly selected friends of } A \text{ are friends}) \\
&= \mathbb{P}(\text{fraction of pairs of } A\text{'s friends that are linked to each other}).
\end{aligned}
$$

Clustering coefficient



- For any given node A and two randomly selected nodes B and C:

$$
\begin{aligned}
CC(A) &= \mathbb{P}(B \in N(C) | B, C \in N(A)) \\
&= \mathbb{P}(\text{two randomly selected friends of } A \text{ are friends}) \\
&= \mathbb{P}(\text{fraction of pairs of } A\text{'s friends that are linked to each other}).
\end{aligned}
$$

- For example, in the Figure above,

node	CC
A	1
B	1/3
C	1/3
D	0
E	0

R code based on package igraph: Clustering coefficient

```
# First generate the graph from edge list
el_cc <- matrix( c("A", "B", "A","C", "B",
"C", "B","E","D","E","C","D"), nc=2, byrow=TRUE)
g_el_cc <- graph.edgelist(el_cc,directed=FALSE)
# Then calculate CC
transitivity(g_el_cc, type="localundirected")
```

Duncan Watts - The Myth of Common Sense

Duncan Watts - The Myth of Common Sense

- http://www.youtube.com/watch?feature=player_detailpage&v=D9XF0QOzWM0

References I