Porto Business School

/ University of Porto

Social Network Analysis

Part 3 – **Basic Concepts and Measurements**

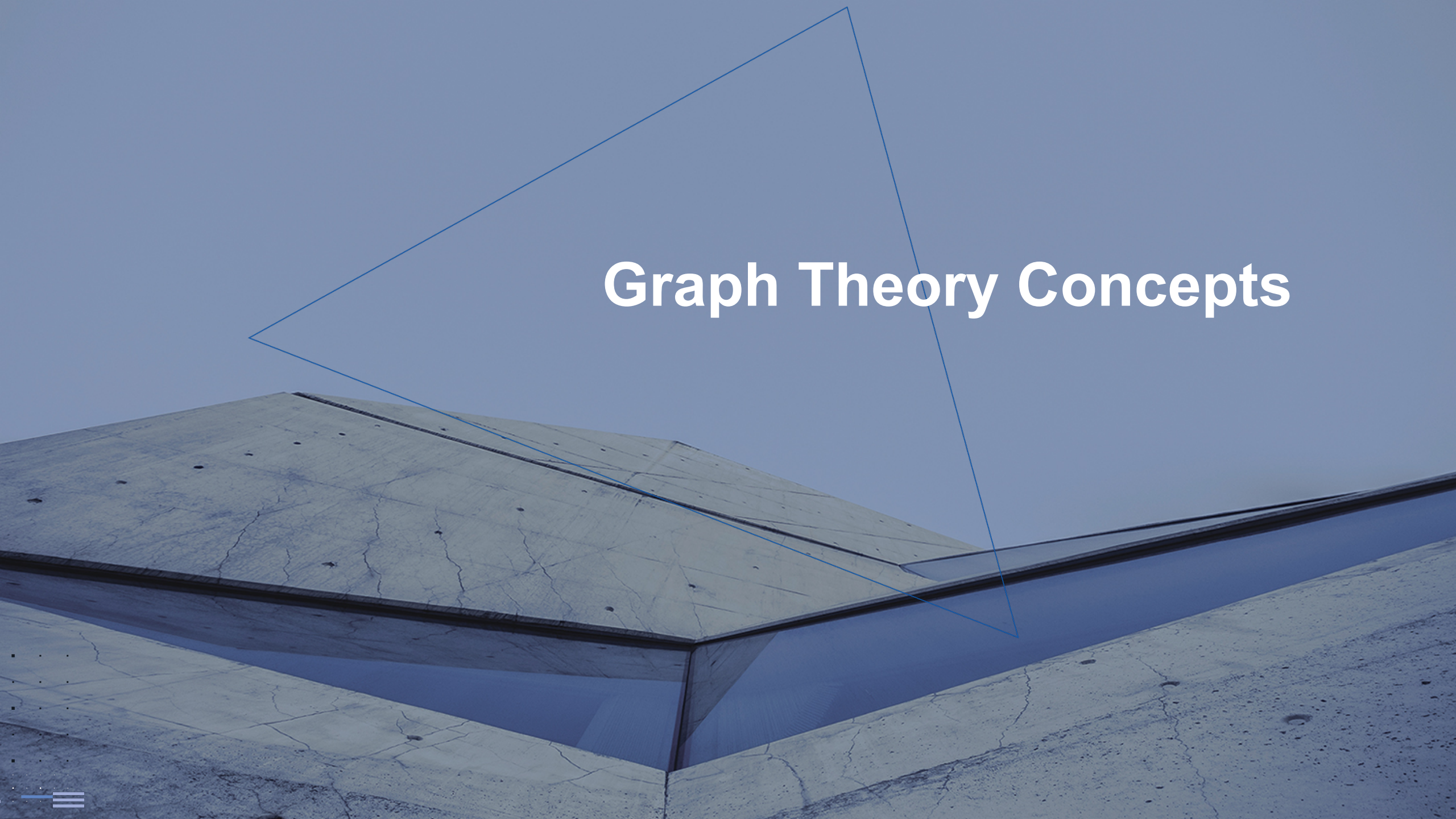
Prof. Eduarda Mendes Rodrigues

ACCREDITATIONS MEMBERSHIPS RANKINGS

Porto Business School
/ University of Porto

Part 3 - Basic Concepts and Measurements

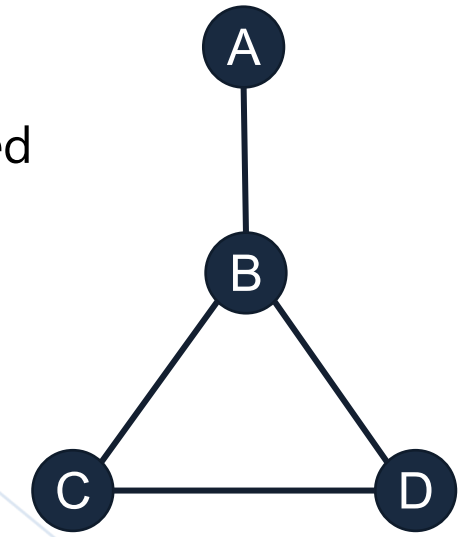
- Graph theory concepts
  - Paths and connectivity
  - Connected components
  - Distance and breadth-first search
- Social network metrics
  - Degree, Clustering coefficient, Cohesion, Density
  - Centrality measures, Clique-census


Graph Theory Concepts

Undirected vs. Directed Graphs

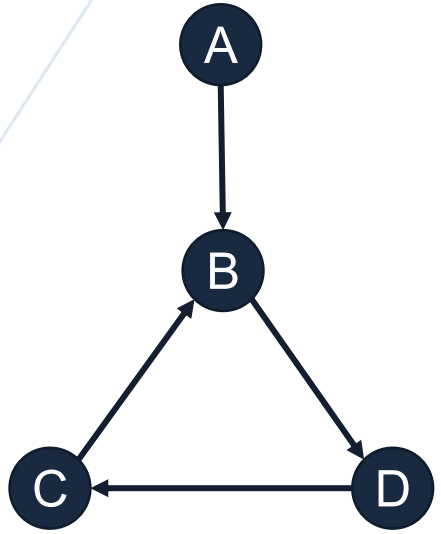
- Graph edges can be **undirected** (symmetric relationship) or **directed** (possibly asymmetric relationship)

Undirected graph



- *A and B are co-workers*
- *C and D read the same book*
- *B and D are friends on Facebook*

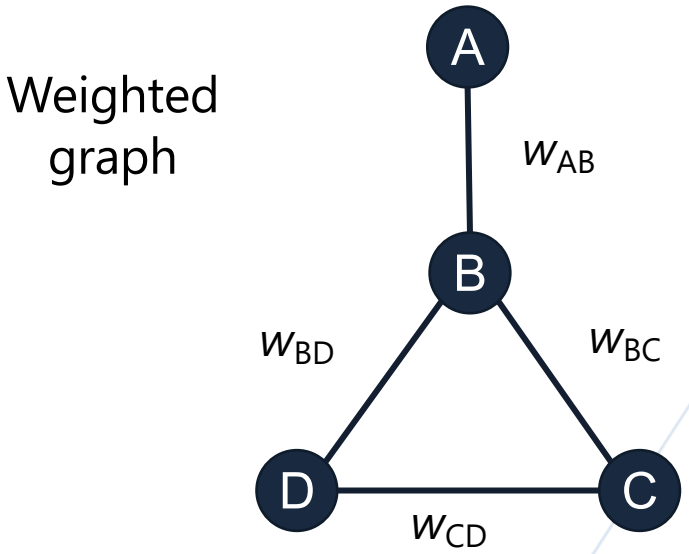
Directed graph



- *A follows B on Twitter*
- *D web page links to B web page*
- *B cites author C*

Weighted Graphs

- Graph edges can be **weighted** (e.g., duration of relationship, number of books, number of citations, frequency of interaction, etc.)



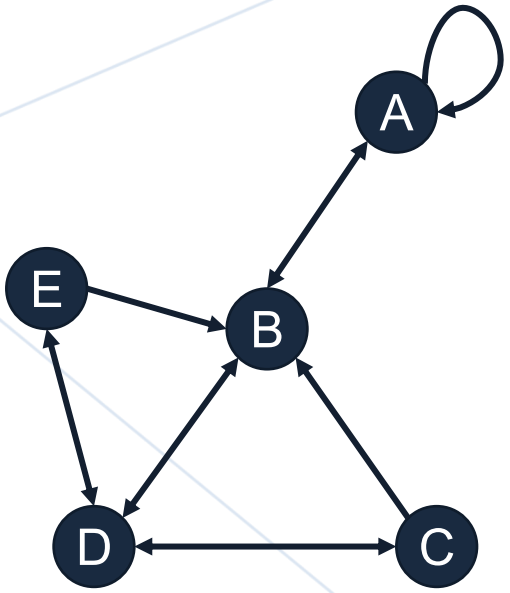
Graph-theoretic Data Structures

- Edge list *: represents edges as a list of adjacent node pairs
- Adjacency list *: represents a list of nodes with a list of adjacent nodes
- Adjacency matrix: represents edges as a data matrix that indicates which nodes are adjacent
  - A_{ij} = 1 if node i has an edge to node j, A_{ij} = 0 otherwise
  - A_{ii} = 0, unless the network has self-loops
  - $A_{ij} = A_{ji}$, if the network is undirected, or if i and j share a reciprocated edge

* More compact representation than matrix. Better for when the network is large and/or sparse.

Graph-theoretic Data Structures

- Graph data can be represented using different data structures



Edge list

A,A
A,B
B,A
B,D
C,B
C,D
D,B
D,C
D,E
E,B
E,D

Adjacency list

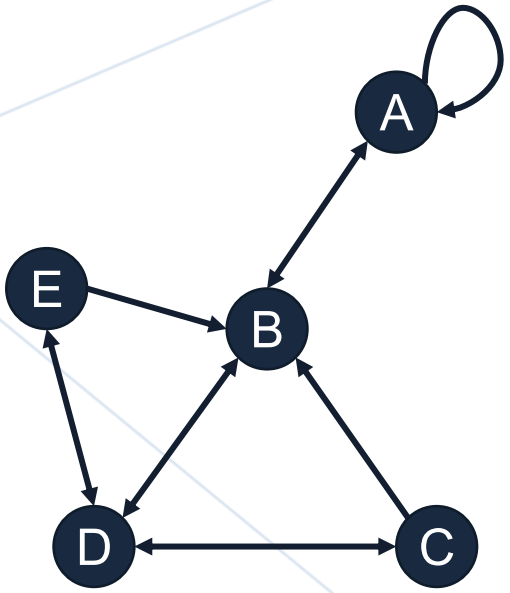
$A \rightarrow A,B$
$B \rightarrow A,D$
$C \rightarrow B,D$
$D \rightarrow B,C,E$
$E \rightarrow B,D$

Adjacency matrix

	A	B	C	D	E
A	1	1	0	0	0
B	1	0	0	1	0
C	0	1	0	1	0
D	0	1	1	0	1
E	0	1	0	1	0

Other Graph Representations

- GraphML (graph markup language) is an XML language



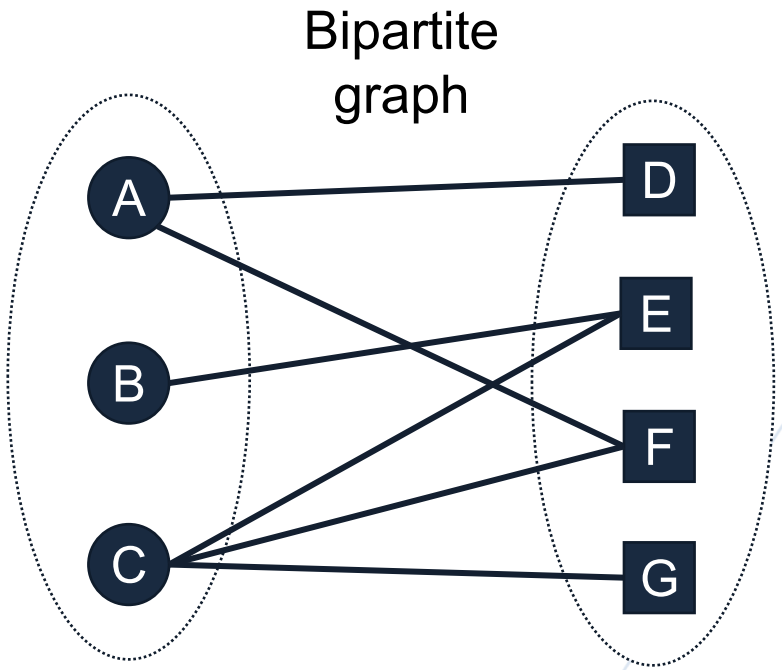
GraphML

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
      http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">

  <graph id="G" edgedefault="directed">
    <node id="A"/> <node id="B"/> <node id="C"/>
    <node id="D"/> <node id="E"/>
    <edge source="A" target="A"/><edge source="A" target="B"/>
    <edge source="B" target="A"/><edge source="B" target="D"/>
    <edge source="B" target="E"/>
    <edge source="C" target="B"/><edge source="C" target="D"/>
    <edge source="D" target="B"/><edge source="D" target="C"/>
    <edge source="D" target="E"/>
    <edge source="E" target="B"/><edge source="E" target="D"/>
  </graph>
</graphml>
```

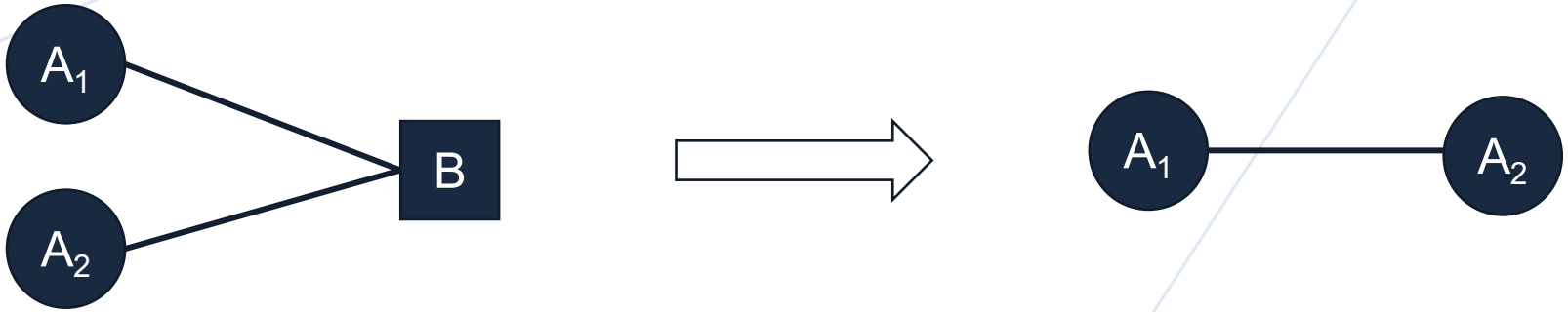
Bipartite Graphs

- Graph may include different types of nodes that form disjoint sets
- No edges between nodes of the same type



Projections

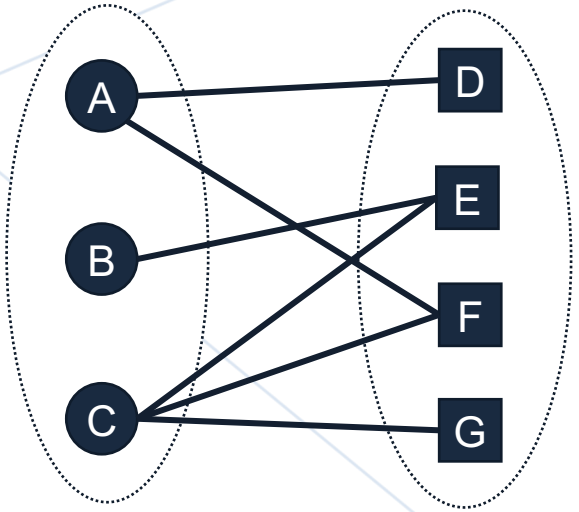
- Implicit networks can be derived from bipartite graphs:



  - **A:** authors, **B:** documents – two people who write the same document are **co-authors**
  - **A:** actors, **B:** movies – two actors co-staring in the same movie are **co-stars**
  - **A:** tags, **B:** images – two tags assigned to the same images **co-occur**

Projections

- A projection is a transformation of a bipartite-network into a single-mode network comprising nodes of a single type

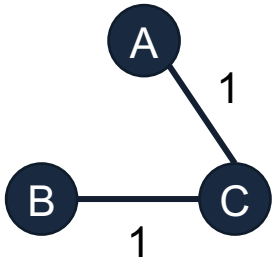


Adjacency matrix A

$$A = \begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{cccc} D & E & F & G \end{array} \\ \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} \end{array}$$

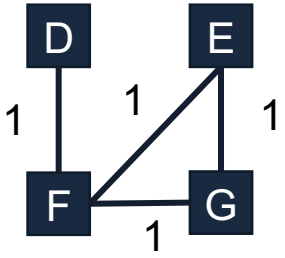
$$A_{circles} = A \cdot A^T$$

$$\begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{bmatrix} 2 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 3 \end{bmatrix} \end{array}$$



$$A_{squares} = A^T \cdot A$$

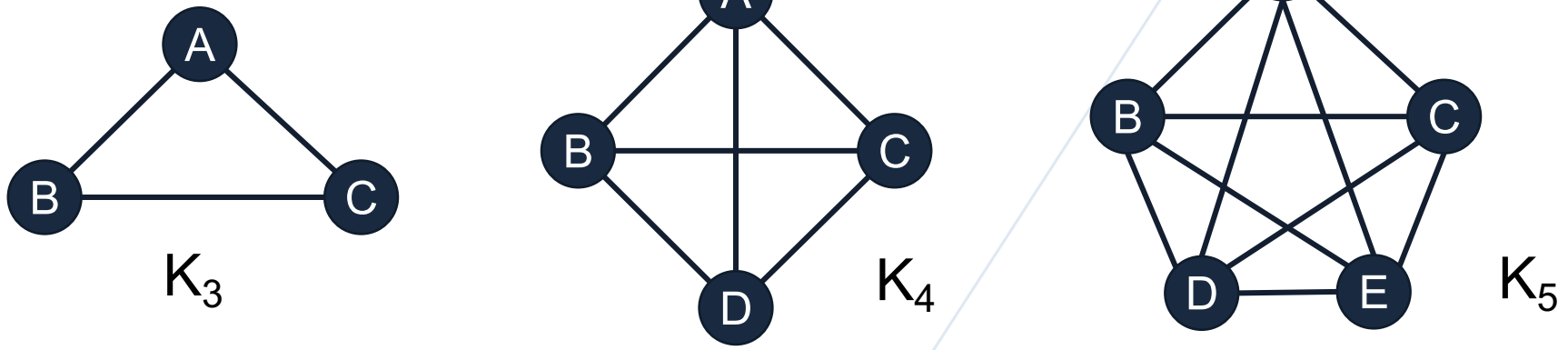
$$\begin{array}{c} \\ D \\ E \\ F \\ G \end{array} \begin{array}{c} \begin{array}{cccc} D & E & F & G \end{array} \\ \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 2 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} \end{array}$$



Complete Graphs / Cliques

- K_n is the **complete graph (clique)** with K vertices
  - each vertex is connected to every other vertex
  - there are n*(n-1)/2 undirected edges

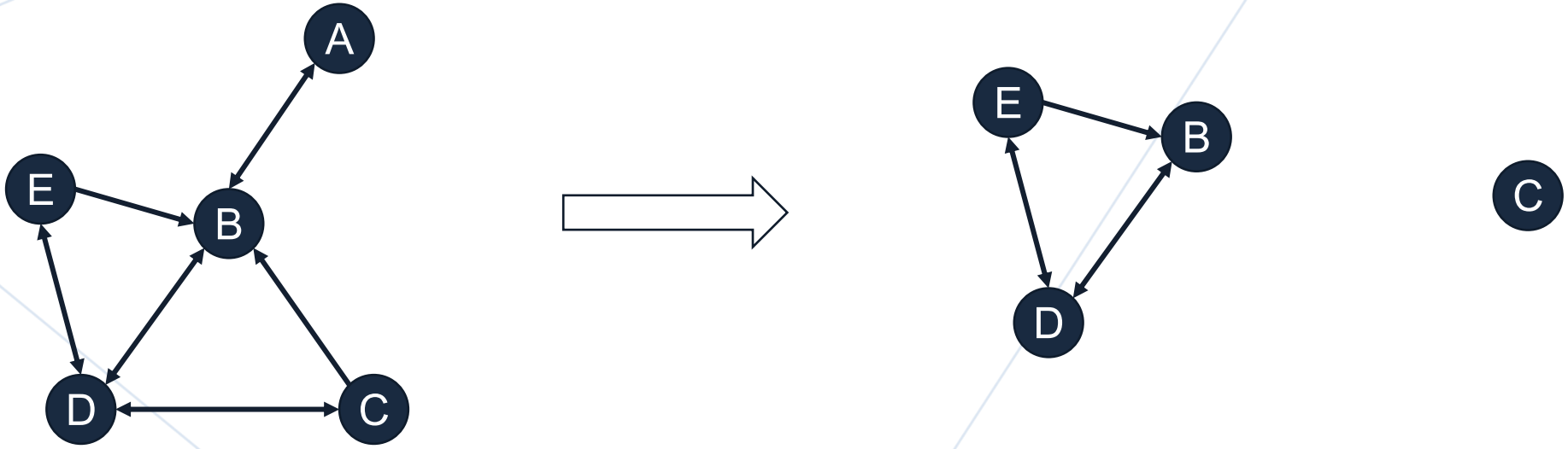
Complete graphs



- **Clique census:** provides statistics about K_n cliques present in the graph

Subgraphs

- Subsets of nodes and edges from a graph are **subgraphs**

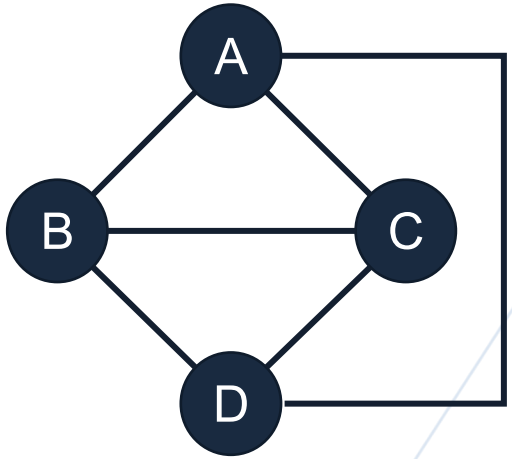


Planar Graphs

- A **planar graph** can be drawn on a place such that no two edges intersect (e.g. Sierpinski triangle)

- How many nodes has the **largest complete graph that is planar**?

Complete
Planar Graph



Planar Graphs

Game: try to **untangle** planar graphs

- Web: https://www.jasondavies.com/planarity/
- App: Untangle Me

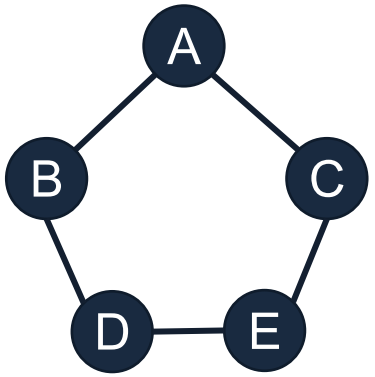
Paths

- A **graph path** is a sequence of nodes with the property that each consecutive pair in the sequence is connected by an edge

- Examples:
  - a passenger taking a sequence of airline flights
  - a piece of information being passed from person to person in a social network
  - a computer user or piece of software visiting a sequence of Web pages by following links

Cycles

- A graph **cycle** is a path with at least three edges, in which the first and last nodes are the same, but otherwise all nodes are distinct



- Example:
  - *Your sister's son's close friend from school is in fact someone who plays basketball with your daughter*
  - This is a cycle consisting of you, your sister, her son, his school friend, his team mate (i.e. your daughter), and finally back to you

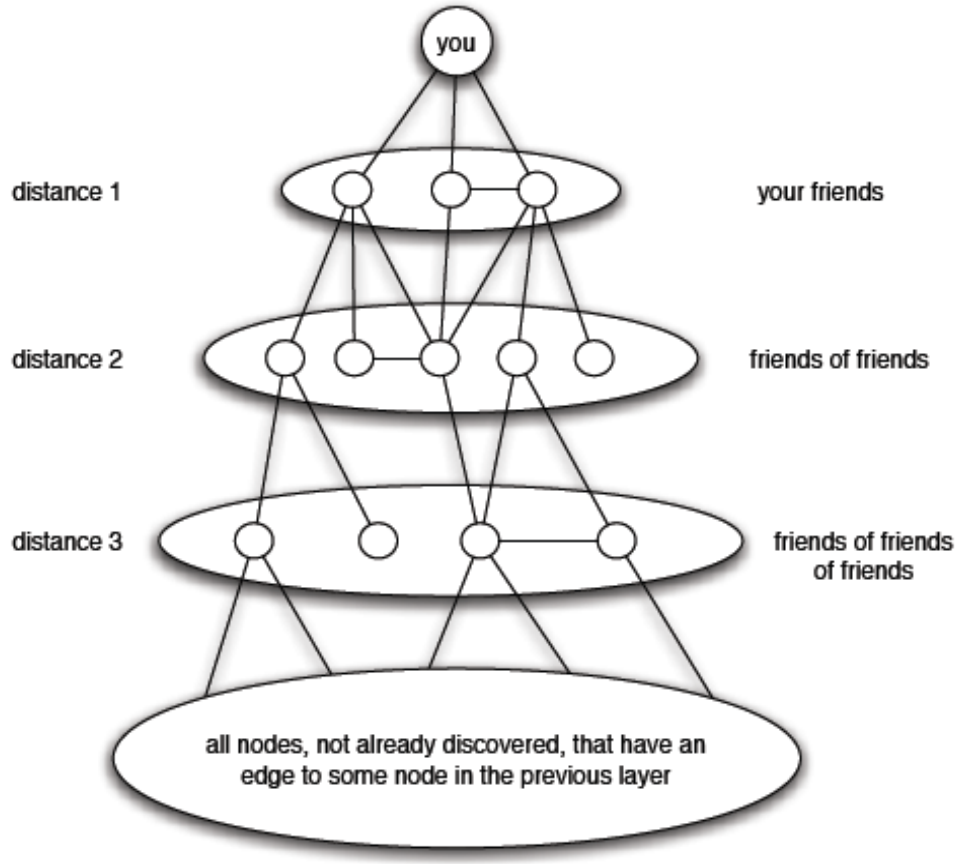
Walks

- A **walk of length L** in a graph is a sequence of L edges (not necessarily different)
  - E.g., AB, BE, ED, DB, …, BD
  - Walk between A and D is denoted by ABEDB…D
- A **walk is closed** if the start node coincides with the end node

Distance and Breadth-First Search

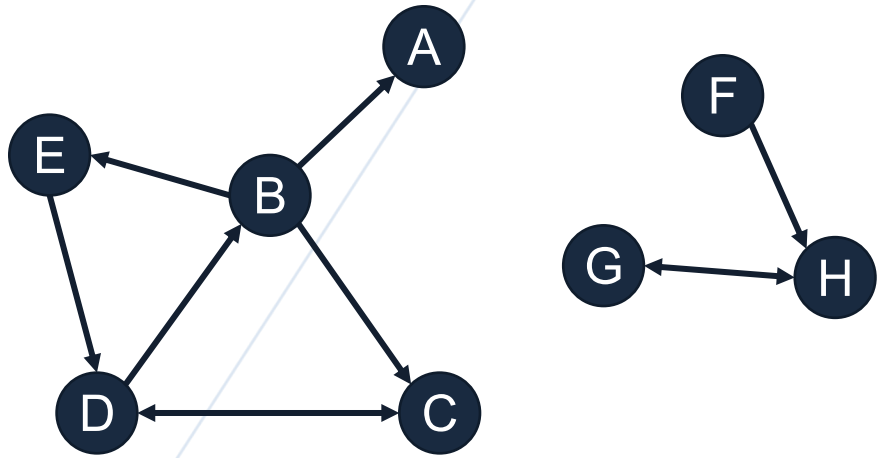
- The **length of a path** is the number of steps it contains from beginning to end, i.e., the number of edges in the sequence that comprises it

- **Breadth-first search** is a technique that allows to determine distances to nodes one layer at a time



Shortest Paths and Graph Diameter

- The **shortest path**, or geodesic distance, between two nodes is the shortest sequence of links connecting them (may not be unique)
- **Eccentricity** of a node: largest geodesic distance between the node and any other node in the graph
- The **graph diameter** is the largest geodesic distance in the graph

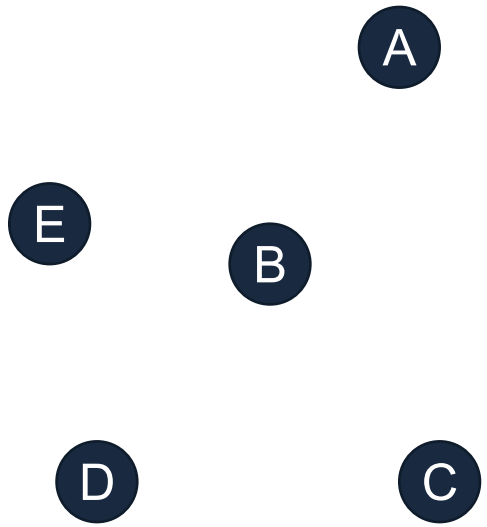


Node	Eccentricity
A	-
B	2
C	3
D	2
E	3
F	2
G	1
H	1

Exercise: Find the eccentricity of each node. What is the shortest path between D and A? Diameter?

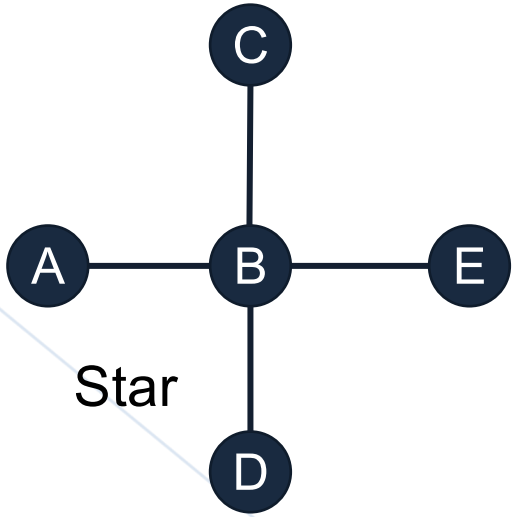
Special Types of Graphs

- **Empty graph** is an edge-less graph

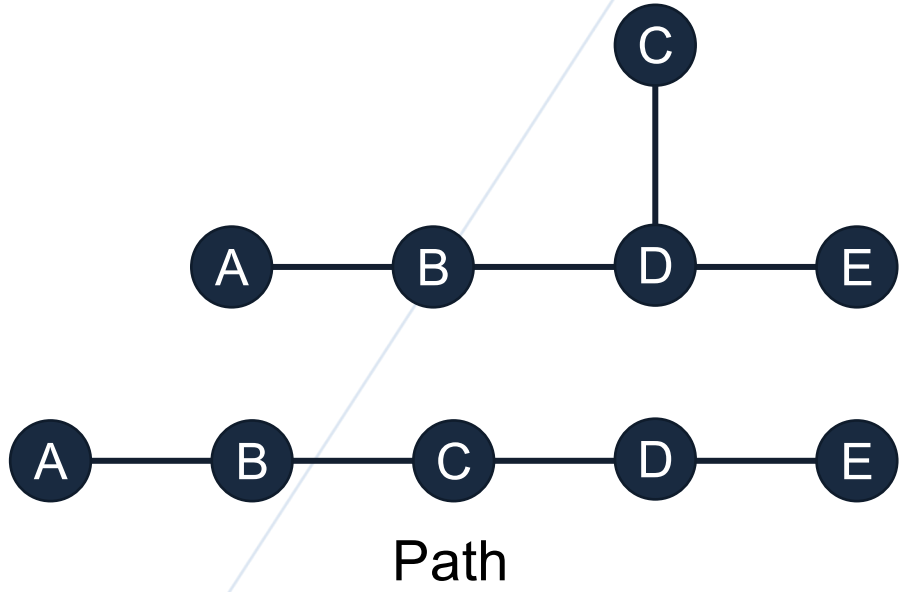


Special Types of Graphs

- **Tree** is an acyclic graph (i.e. no cycles exist)
- Two nodes have exactly one path between them



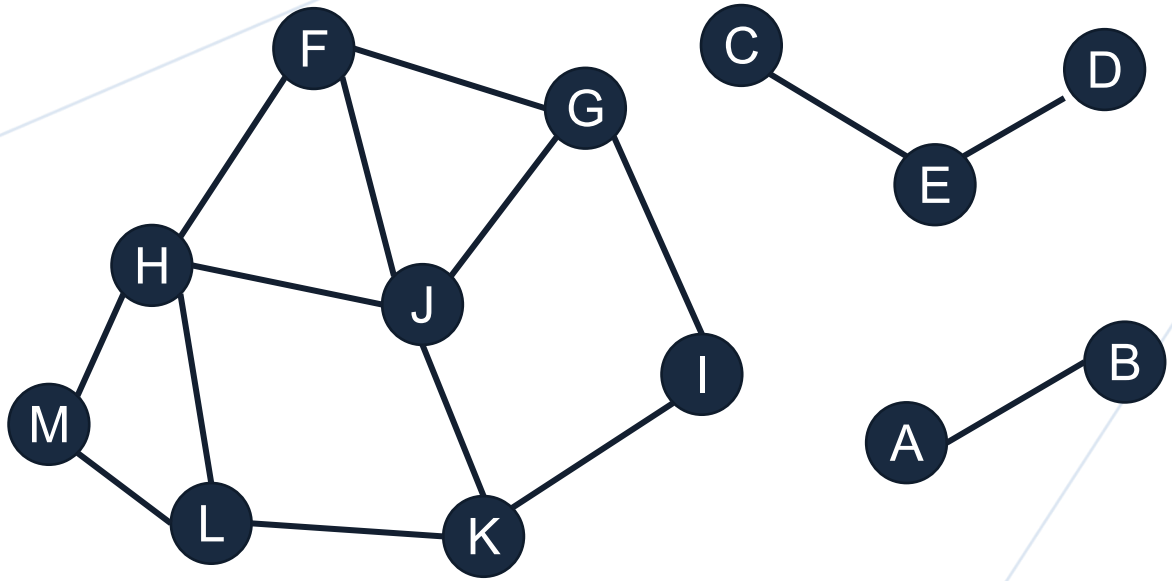
Star



Path

Graph Connectivity

- A graph is **connected** if for every pair of nodes, there is a path between them

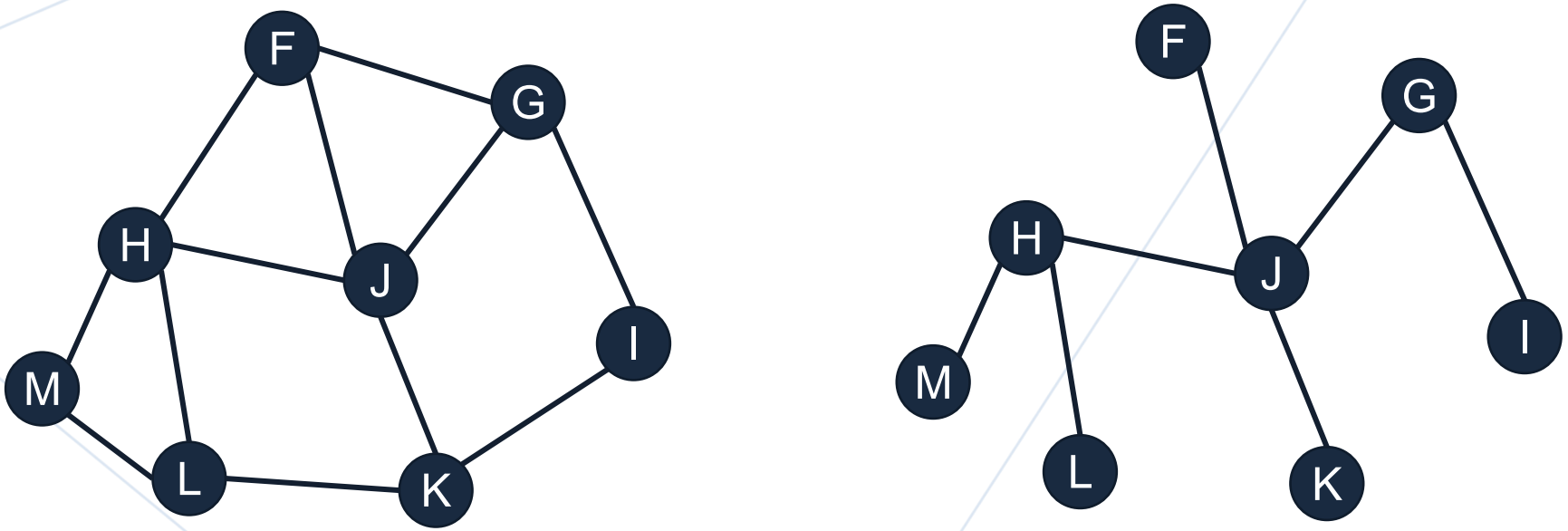


Example of a disconnected graph with 3 connected components

- A **connected component** of a graph is a subset of the nodes where
  - every node in the subset has a path to every other;
  - the subset is not part of some larger set with the property that every node can reach every other

Spanning Tree of a Graph

- A **spanning tree** of a connected graph G is a **subgraph** that includes all the nodes of G and **is a tree**

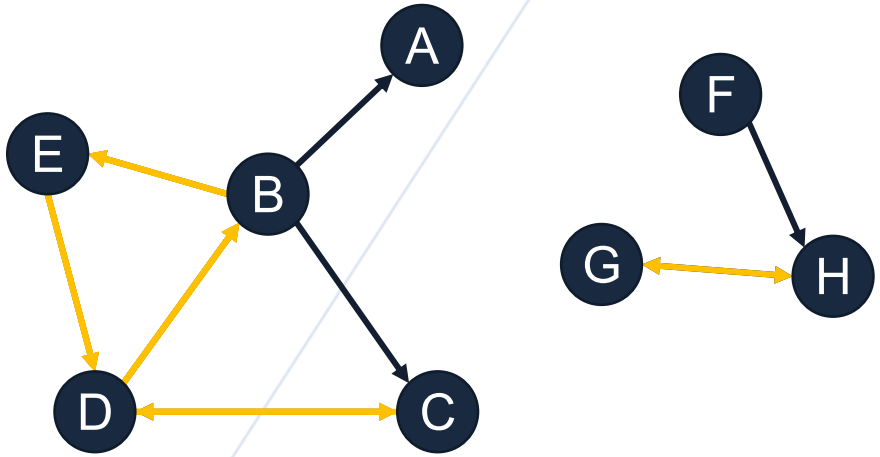


A graph and one possible spanning tree

Strong and Weak Components

- Each node within a **strongly connected component** (SCC) can be reached from every other node in the component by following directed links

- Each node within a **weakly connected component** (WCC) can be reached from every other node in the component by following links in either direction



WCC:
- A, B, C, D, E
- F, G, H

SCC:
- A
- B, C, D, E
- F
- G, H

Exercise: List all the WCCs and SCCs of this network.

Giant Component

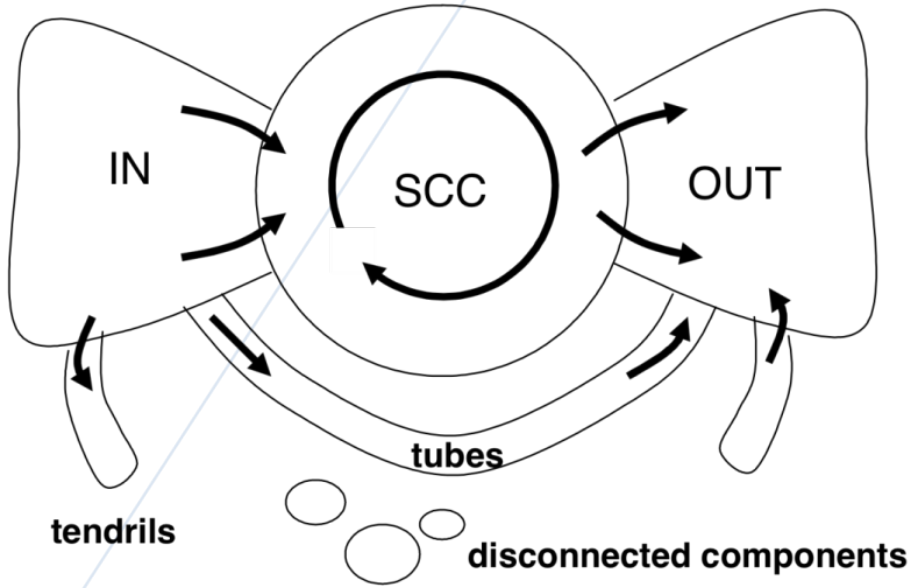
- A **giant component** is a connected component that contains a significant fraction of all the nodes

- When a network contains a giant component, almost always contains a single one

- All it takes is a single edge to connect smaller components to the giant component!

Bow-tie Model of the Web

- The web is a directed graph: web pages link to other pages
- The connected components tell us what set of pages can be reached from any other just by surfing (no ‘jumping’ around by typing in a URL or using a search engine)
- Broder et al. (2000) – crawl of over 200 million pages and 1.5 billion links



SCC – 27.5%

IN and OUT – 21.5%

Tendrils and tubes – 21.5%

Disconnected – 8%

Try it out in R: Create Graph

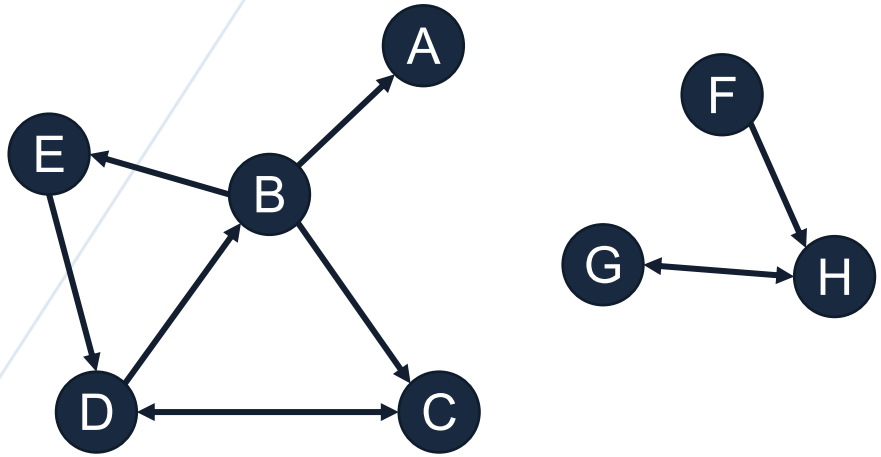
```
library(igraph)

# Create graph

g <- graph.empty(directed = TRUE)
g <- g + vertices('A','B','C','D','E','F','G','H')

# Add edges

g <- g + edges('B','A')
g <- g + edges('B','E')
g <- g + edges('B','C')
g <- g + edges('C','D')
g <- g + edges('D','B')
g <- g + edges('D','C')
g <- g + edges('E','D')
g <- g + edges('F','H')
g <- g + edges('H','G')
g <- g + edges('G','H')
```



Try it out in R: Connected Components

Porto Business School
/ University of Porto

```
library(igraph)

# Create graph

# Compute connected components
wcc=clusters(g, mode = c("weak"))
scc=clusters(g, mode = c("strong"))

# Define colors for plotting
colors <- rainbow(10)
V(g)$color <- colors[scc$membership] # change scc by wcc
V(g)$size <- 60
plot(g)
```

Try it out in R: Shortest Paths

Porto Business School / University of Porto

```
# Compute shortest path

sp = get.shortest.paths(g, from="E", to="A", output="both")

# Define colors for plotting

V(g)$color <- "white"
E(g)$color <- "gray"
V(g)[sp$vpath[[1]]]$color <- "red"
E(g)[sp$epath[[1]]]$color <- "red"
E(g)$arrow <- 2
V(g)$size <- 30
plot(g)


# Compute shortest path between other pairs of nodes
# What happens when nodes belong to different components?
```

Try it out in R: Diameters, Cliques…

Porto Business School
/ University of Porto

```
# Compute the graph diameter

d = diameter(g, directed = TRUE, unconnected = TRUE, weights = NULL)
print(d)

# Compute size of largest clique(s)

cliques_n <- clique.number(g) # ignores directionality
print(paste("Size of largest clique(s):",cliques_n))

# Get largest cliques

l_cliques <- largest.cliques(g)
print(paste("Largest clique:", l_cliques))
```

Try it out in R: k-Cliques

Porto Business School
/ University of Porto

```
# Get cliques by size

colors <- rainbow(10)
K3 <- cliques(g, min=2)
E(g)$color <- "gray"

for (i in 1:length(K3)) {
  V(g)$color <- "white"
  V(g)[K3[[i]]]$color <- colors[i]
  plot(g)
  readline()
}
```

Social Network Metrics

Social Network Metrics

- Social network graphs can be analysed using a number of **metrics** including:
  - **cohesion** of the network or sub-network
    measures the ease with which connections can be made
  - **density** of the network or sub-network
    measures the robustness of the connections
  - **centrality** of the nodes
    gives a rough indication of the social power of a node in the network
    - degree, betweenness, closenness, eigenvector (PageRank), network centralization

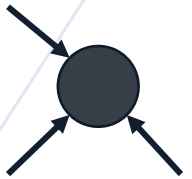
Degree Centrality

- Conveys the **number of links to / from** other nodes in the network
- Higher degree of a node might indicate that the node is a **hub in the network**
- **Most connected does not mean most powerful!**

© David Ramos / Getty Images

Degree Centrality (Prestige)

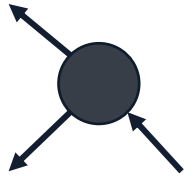
- Property of the nodes derived from immediate connections
  - **In-degree**
    how many directed edges are incident on a node



$$\sum_{i=1}^{n} A_{ij}$$

in-degree=3

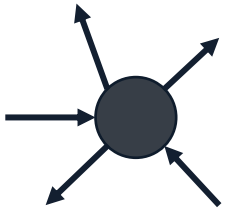
  - **Out-degree**
    how many directed edges originate at a node



$$\sum_{j=1}^{n} A_{ij}$$

out-degree=2

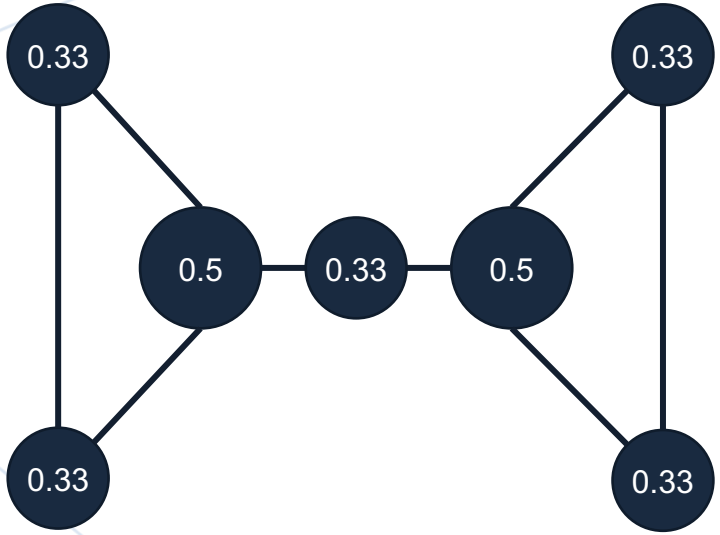
  - **(Total) degree**
    number of edges incident on a node (in or out)

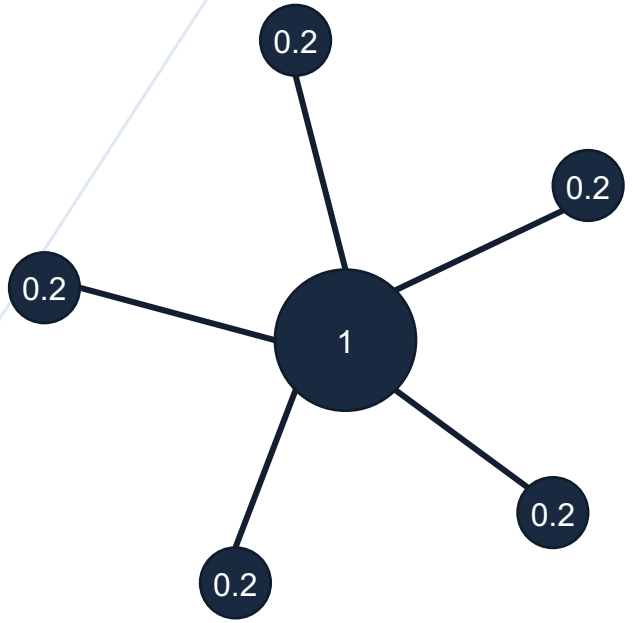


degree=5

Normalized Degree Centrality

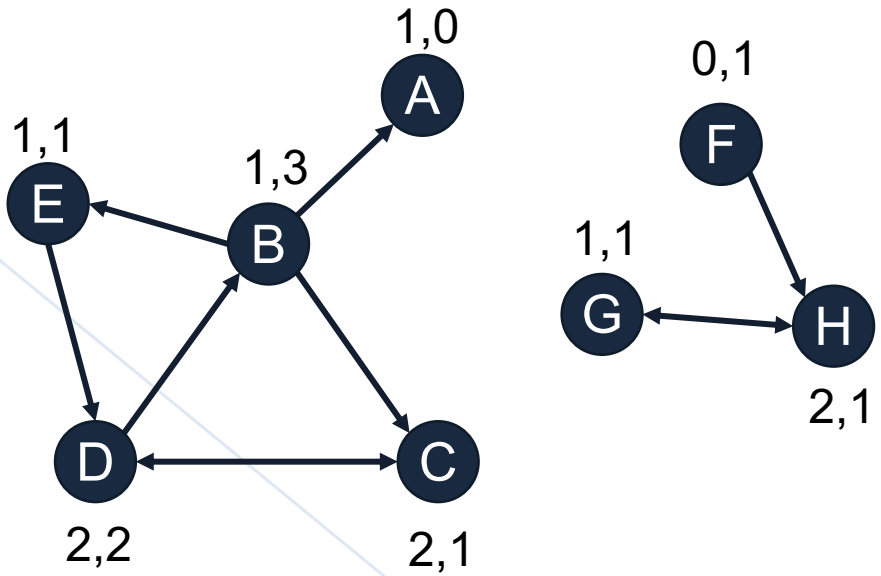
- Divide by the max. possible, i.e. (n-1)

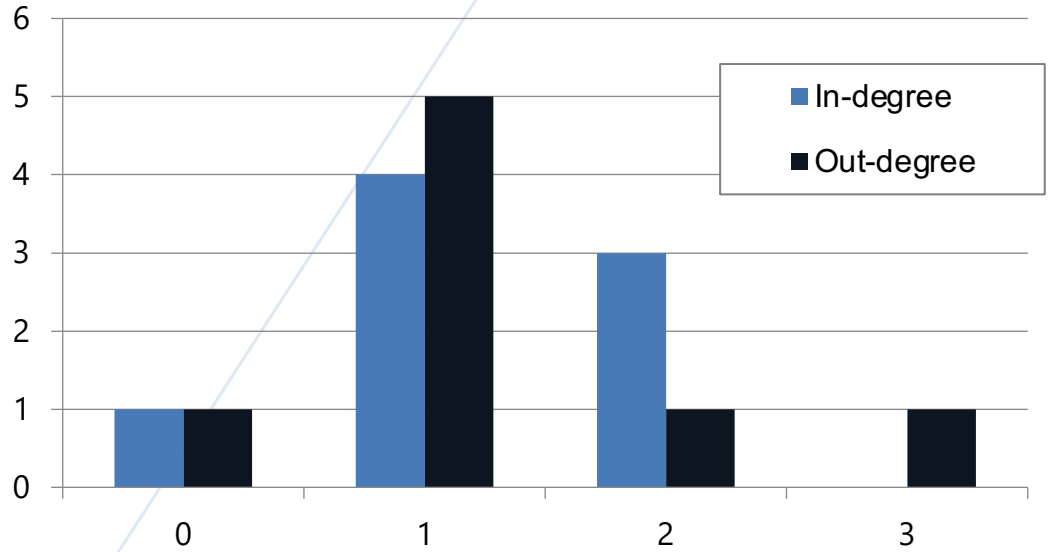




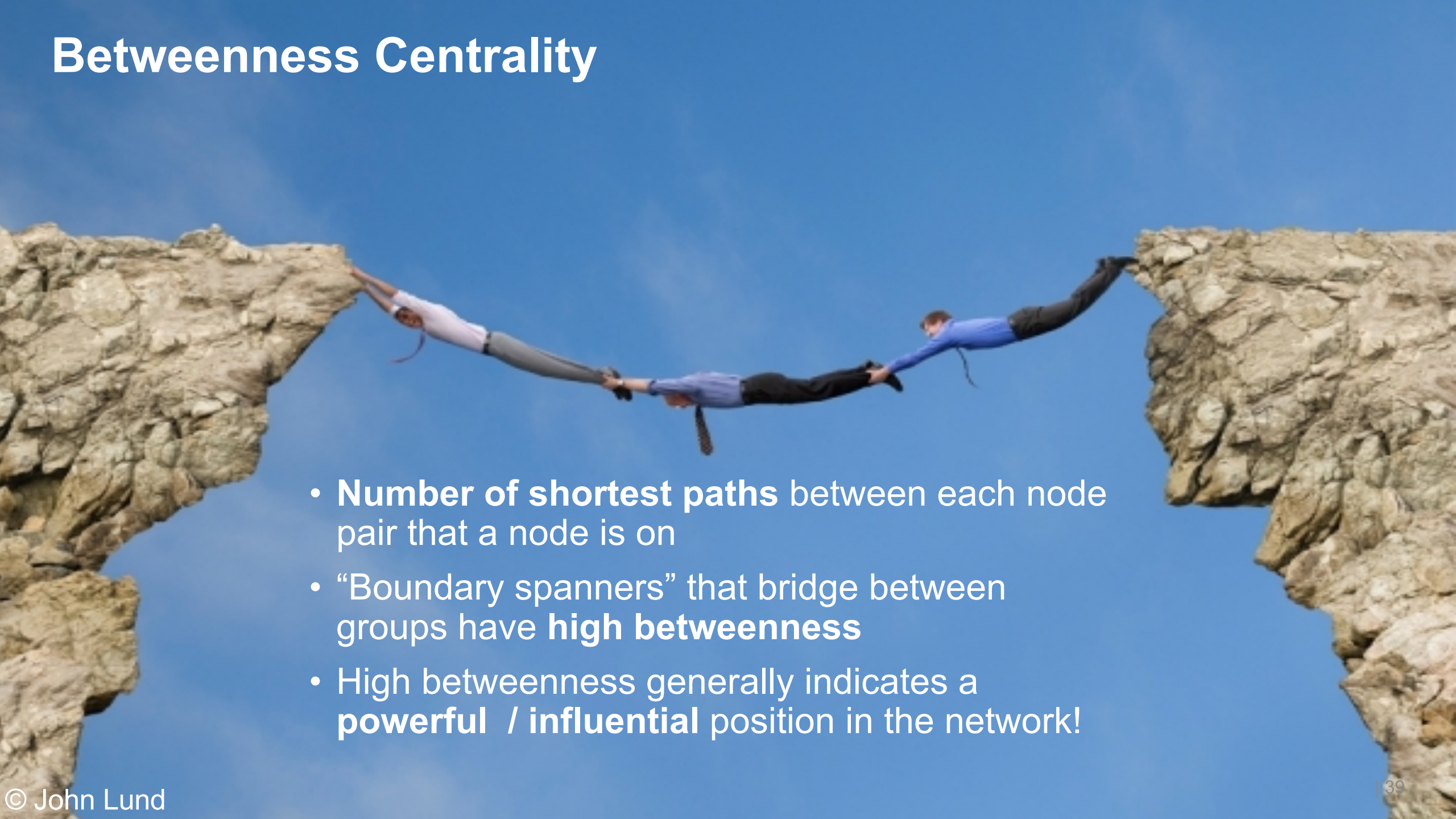
Degree Distribution

- The degree distribution of a graph is a frequency count of the occurrence of each degree





Betweenness Centrality



- **Number of shortest paths** between each node pair that a node is on
- "Boundary spanners" that bridge between groups have **high betweenness**
- High betweenness generally indicates a **powerful / influential** position in the network!

© John Lund

Betweenness Centrality

- Definition: $$C_B(i) = \sum_{j<k} g_{jk}(i) / g_{jk}$$

- Where, g_{jk} = the number of geodesic distances connecting *j*-*k*, and

  g_{jk} (i) = the number that node *i* is on.

- Normalised betweeness centrality:

$$C_B'(i) = C_B(i) / \underbrace{[(n-1)(n-2)/2]}$$

number of pairs of vertices excluding the vertex itself

Closeness Centrality

- Inverse of **the mean shortest path** between a node and all other nodes in the network reachable from it
- Reflects the **ability of a node in accessing information** through the network
- Low closeness generally indicates **high visibility** of what's going on in the network!

© Will Ockenden

Closeness Centrality

- Definition:

$$C_c(i) = \left[\sum_{j=1}^{N} d(i,j) \right]^{-1}$$

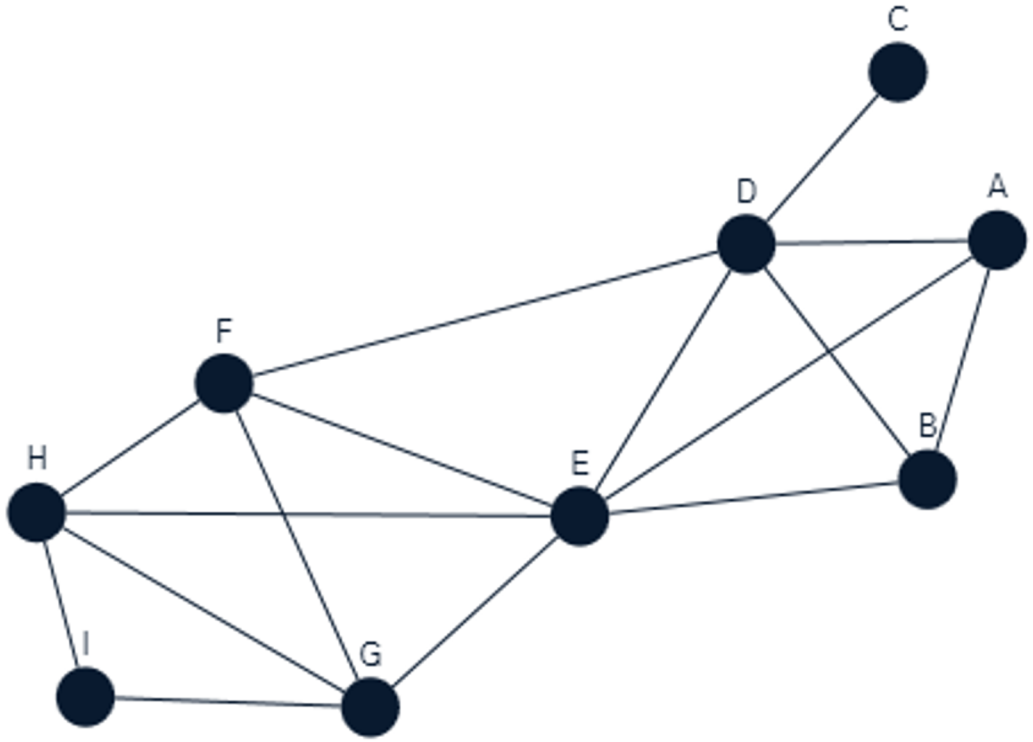
- Normalized closeness centrality:

$$C'_C(i) = (C_C(i)) / \underbrace{(N-1)}$$

number of nodes
excluding the current node

Centrality Mesures and Social Network Roles

- **Peripheral** – below average centrality (C)
- **Central connector** – above average centrality (D)
- **Broker** – above average betweenness (E)



Eigenvector Centrality / PageRank

Porto Business School / University of Porto

- The centrality of the nodes depend on the centrality of the adjacent nodes, and so forth
- Is a recursive **measure of influence** in a network
- Largest positive **eigenvalues** of the graph **adjacency matrix** (concepts from linear algebra)
- Google's ranking algorithm **PageRank** is a variant of the eigenvector centrality measure

Network Centralization (L. Freeman's approach)

- This metric intends to quantify **how equal are the nodes** in the network (using some centrality measure C_x)

- Expresses the **degree of inequality or variance** in the network as a sum of distances in centrality between the most central node and all other nodes, normalised by the maximum sum of distances

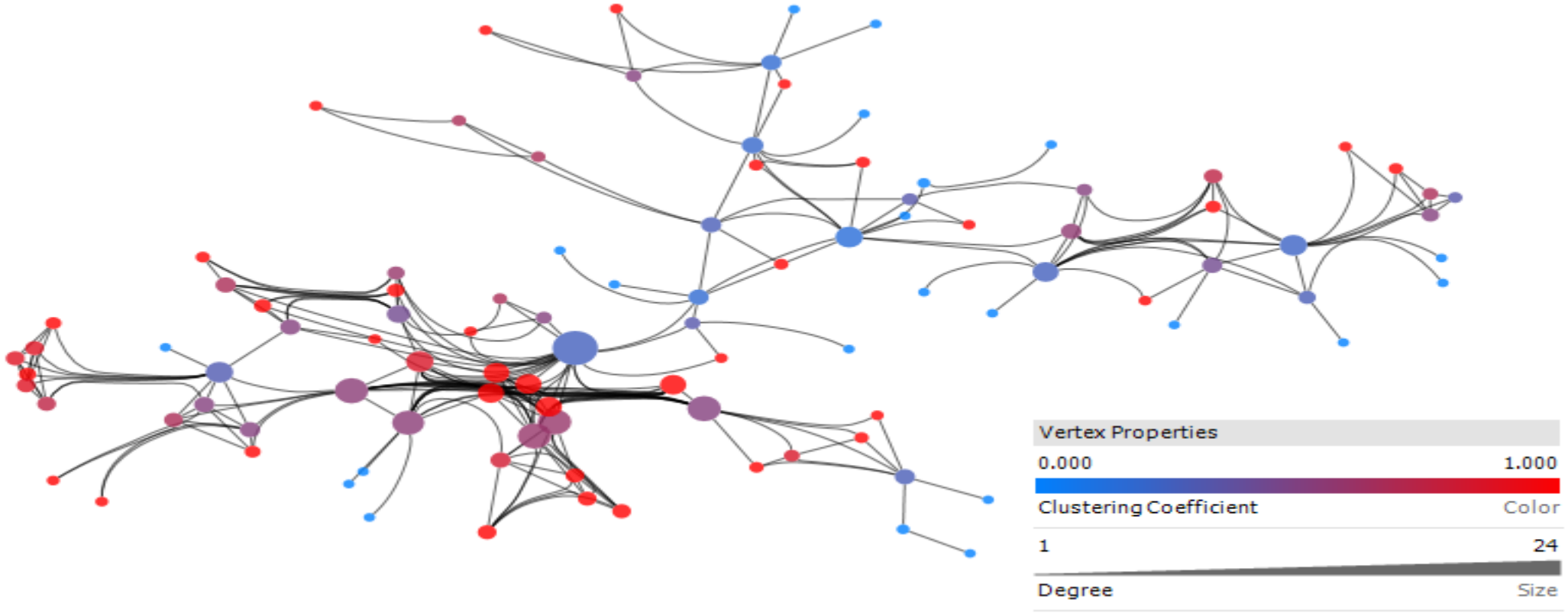
$$C_x = \frac{\sum_{i=1}^{N} C_{x\_max} - C_{xi}}{max \sum_{i=1}^{N} C_{x\_max} - C_{xi}}$$

Density and Clustering Coefficient

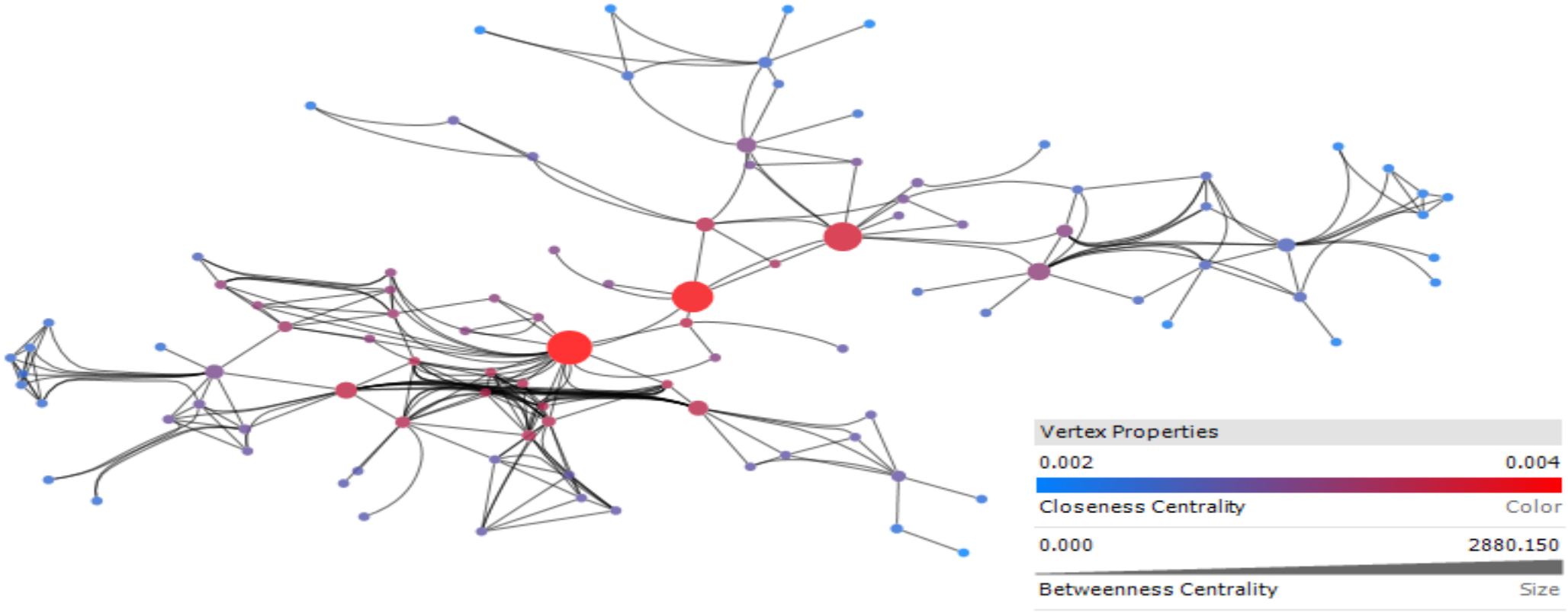
- The **graph density** measures the fraction of all the connections that may exist between among N that are in fact present in the graph
  - *directed graph* - $\mathbf{e_{max} = N*(N-1)}$, i.e., each of the N nodes can connect to (N-1) other nodes
  - *undirected graph* - $\mathbf{e_{max} = N*(N-1)/2}$, i.e., since edges are undirected, count each one only once

- The **clustering coefficient** is a node metric that quantifies the density of the subgraph that includes only the neighbors of the node
  - Would these measures be useful for comparing networks of different sizes (different numbers of nodes)?

Example: Clustering Coefficient & Degree



Example: Betweeness & Closeness



Try it out in R: Network Metrics

Porto Business School / University of Porto

- Calculate metrics on US Senators and Les Misérables graphs

```
# Calculate metrics

# Degree
V(g)$degree     <- degree(g, v=V(g), mode="total", loops = FALSE)
V(g)$indegree   <- degree(g, v=V(g), mode="in", loops = FALSE)
V(g)$outdegree  <- degree(g, v=V(g), mode="out", loops = FALSE)

# Nearest neighbours
V(g)$knn        <- graph.knn(g, V(g))$knn

# Clustering coefficient
V(g)$clust_coeff = transitivity(g, type="local",vids=V(g))

# Other centrality measures
V(g)$betweenness = betweenness(g, v=V(g), directed=TRUE)
V(g)$closeness = closeness(g, v=V(g), mode="all")
V(g)$eigenvector = evcent(g, scale = TRUE)$vector
V(g)$pagerank = page.rank(g, vids = V(g), directed = TRUE, damping = 0.85)$vector
```

References

Bibliography

- Richard. J. Trudeau (2003). Introduction to Graph Theory. Dover Publications.
- R. Diestel (2010). Graph Theory. Springer-Verlag, Heidelberg.
- Hanneman, R. A. & Riddle, M. (2005). Introduction to social network methods. Riverside, CA: University of California, Riverside.
- Hansen, D.L, Schneiderman, B. & Smith, M. (2011). Analyzing Social Media Networks with NodeXL: Insights from a Connected World. New York: Morgan Kaufmann.
- Wasserman, S. & Faust, K. (1994). Social network analysis: Methods and applications. New York: Cambridge University Press.
- Scott, John (2000). Social Network Analysis: A Handbook. London: SAGE Publications Ltd.

References

Publications

- Broder, Andrei; Ravi Kumar; Farzin Maghoul; Prabhakar Raghavan; Sridhar Rajagopalan; Raymie Stata; Andrew Tomkins; Janet Wiener (2000). Graph structure in the web. Proc. of the 9th WWW Conference.
- Freeman, Linton C (1979). "Centrality in social networks conceptual clarification." Social networks 1.3 : 215–239.
- Page, Lawrence; Brin, Sergey; Motwani, Rajeev andWinograd, Terry (1999). "The PageRank citation ranking: Bringing order to the Web". Google Technical Report

Make change happen