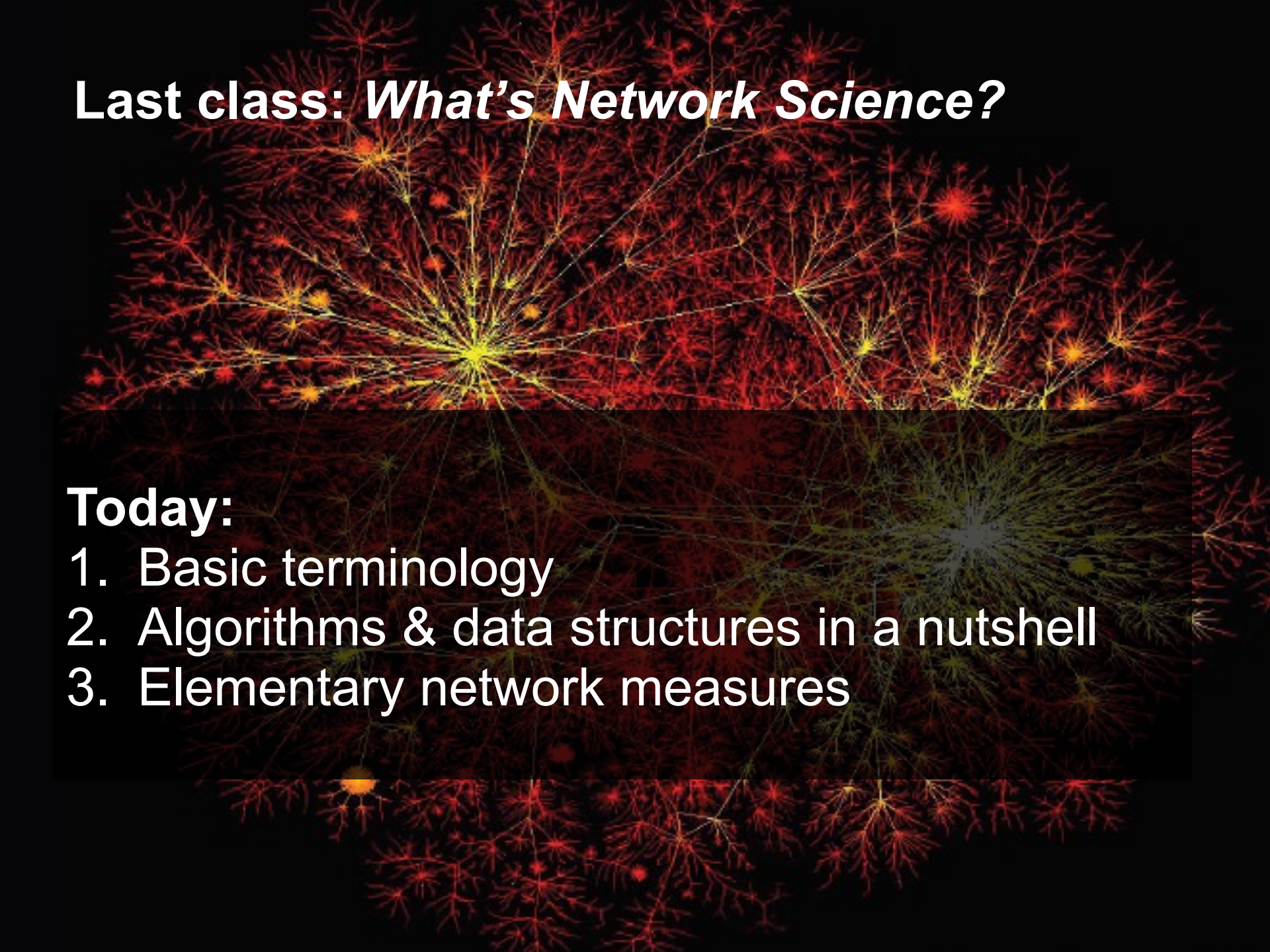
DEI
DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
TÉCNICO LISBOA

Introduction to graphs, algorithms & network metrics

Network Science, 2023/2024



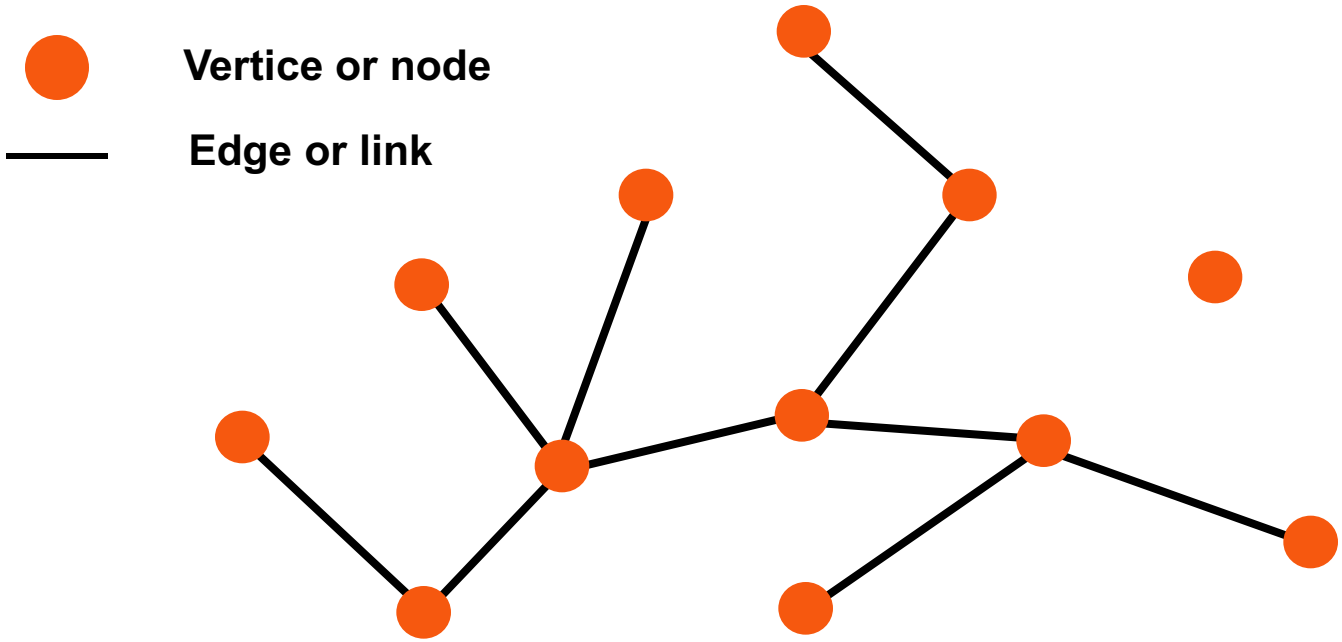
Last class: *What's Network Science?*

Today:

1. Basic terminology
2. Algorithms & data structures in a nutshell
3. Elementary network measures

Networks or graphs - Definition

- Set *N* of nodes (or vertices) and *E* edges or links
  - Edges connect nodes
  - Each node may be connected with any other vertice.

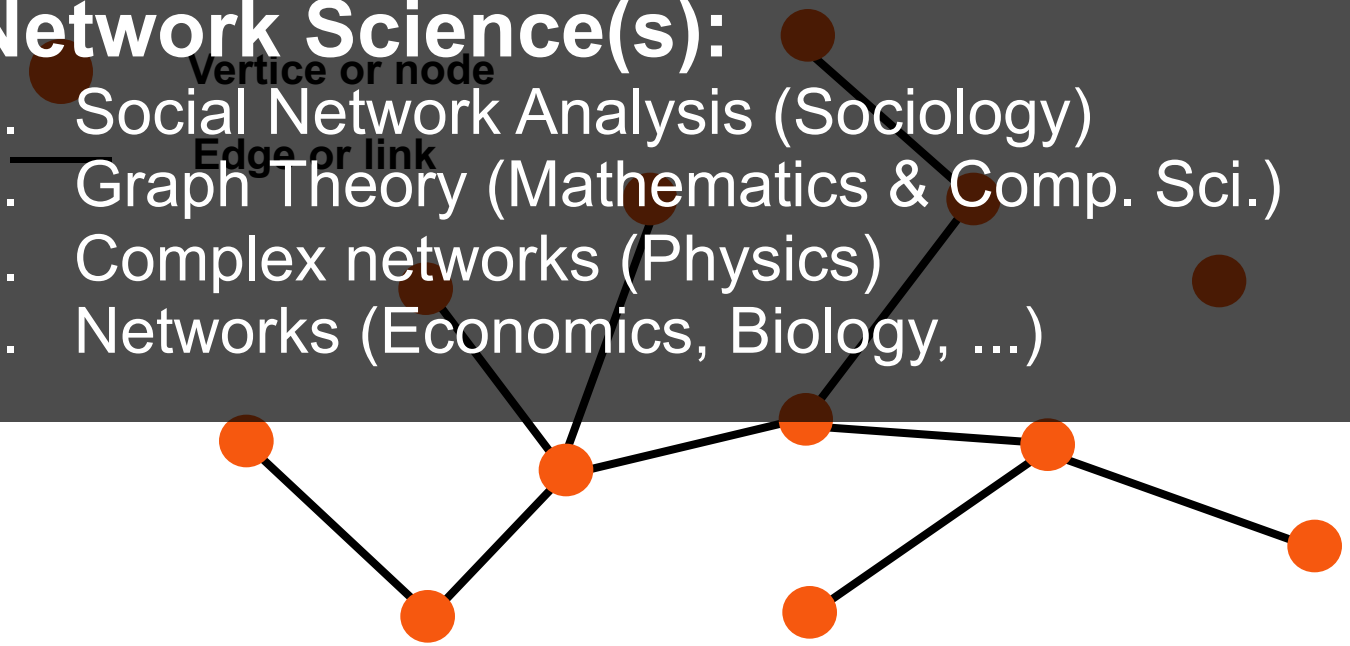


Networks or graphs - Definition

- Set *N* of nodes (or vertices) and *E* edges or links
  - Edges connect nodes
  - Each node may be connected with any other vertice.

Network Science(s):

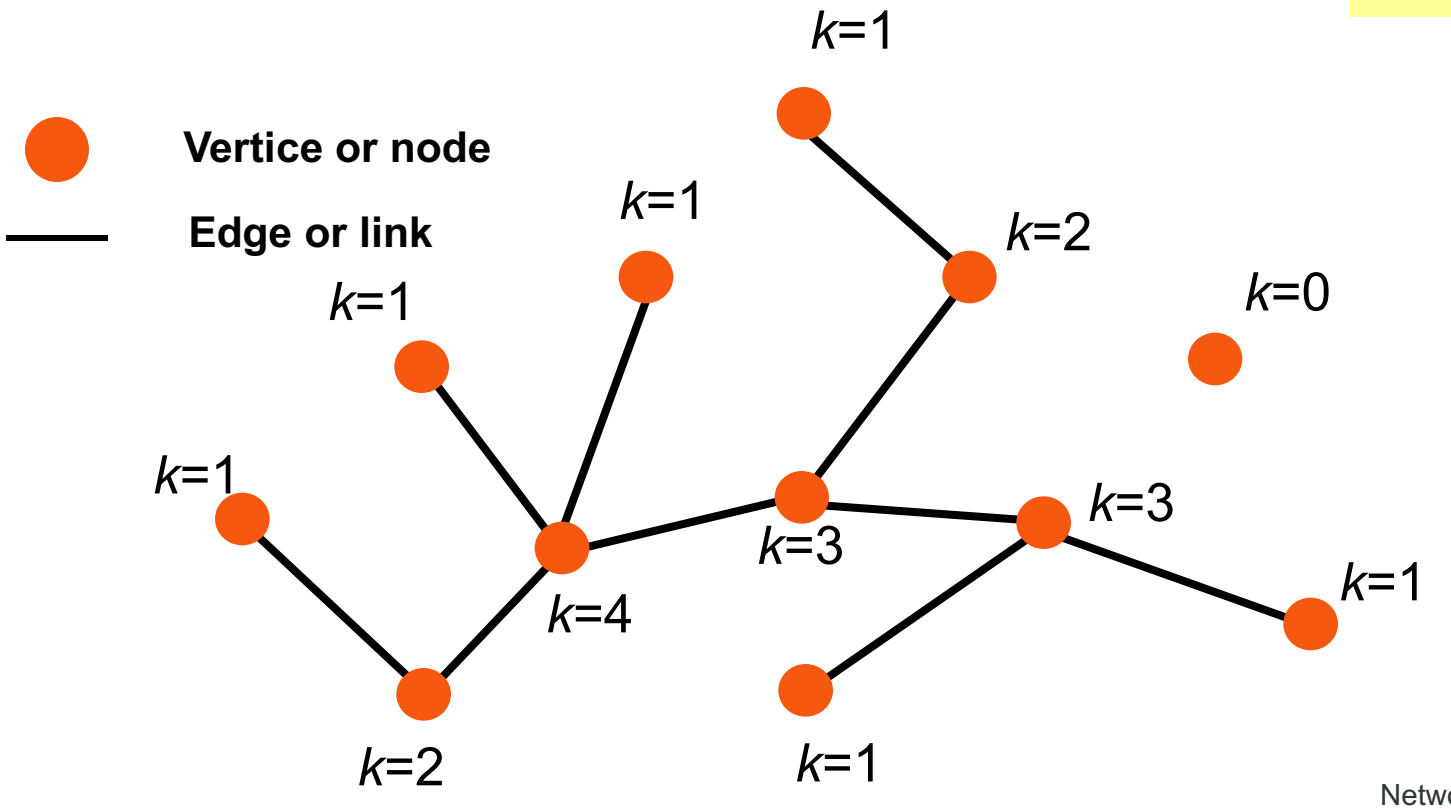
1. Social Network Analysis (Sociology)
2. Graph Theory (Mathematics & Comp. Sci.)
3. Complex networks (Physics)
4. Networks (Economics, Biology, ...)



Degree of a node

- The degree *k* of a node is the number of edges connected to it.
- The average degree of a network is given by

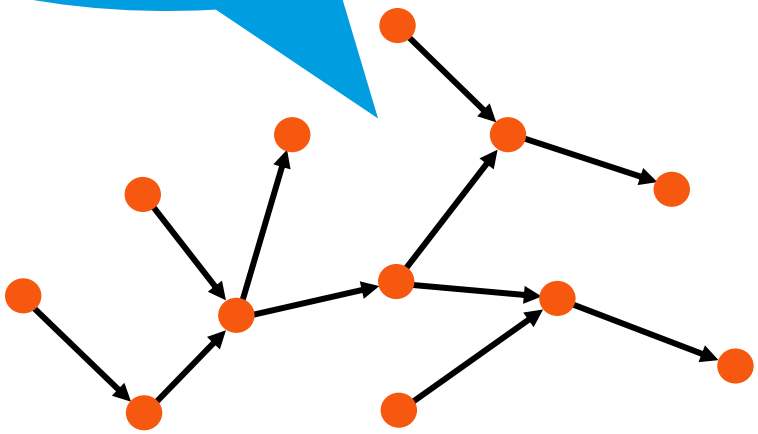
$$z \equiv \langle k \rangle = \frac{2E}{N}$$



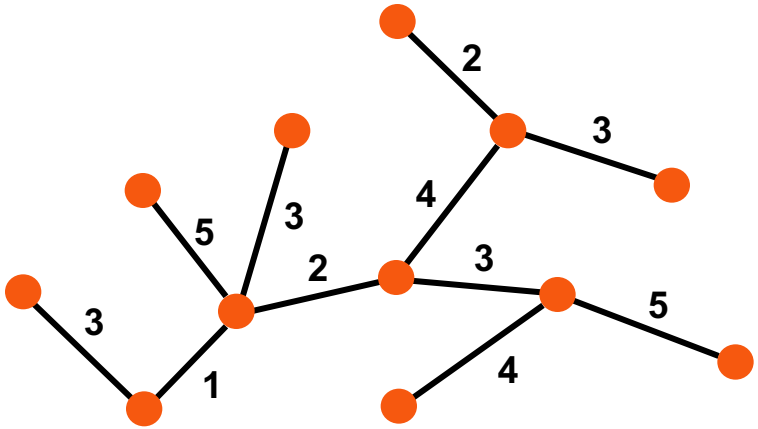
Other definitions

We get a in-degree and a out-degree

- ***Directed graph***
  - where the edges have a direction associated with them.

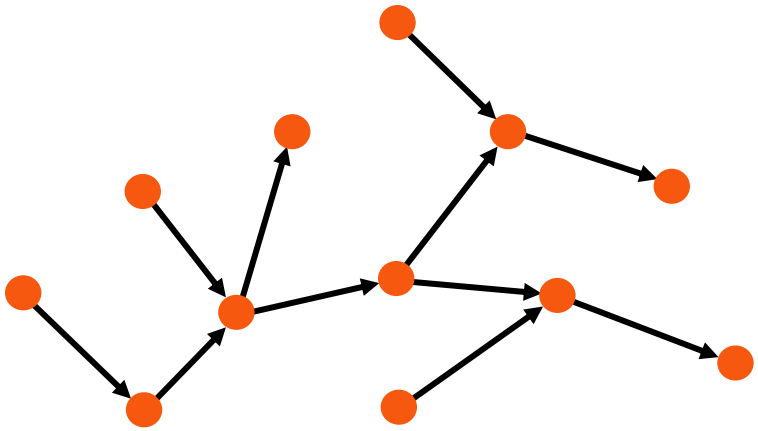


- ***Weighted graph***
  - where the ties among nodes have weights assigned to them.

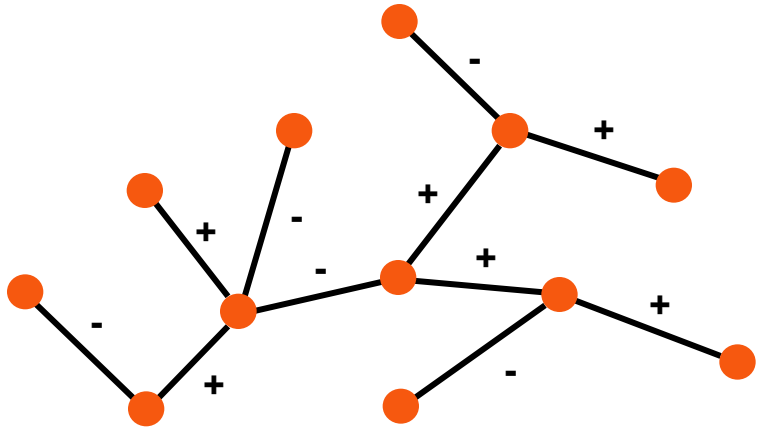


Other definitions

- ***Directed graph***
  - where the edges have a direction associated with them.



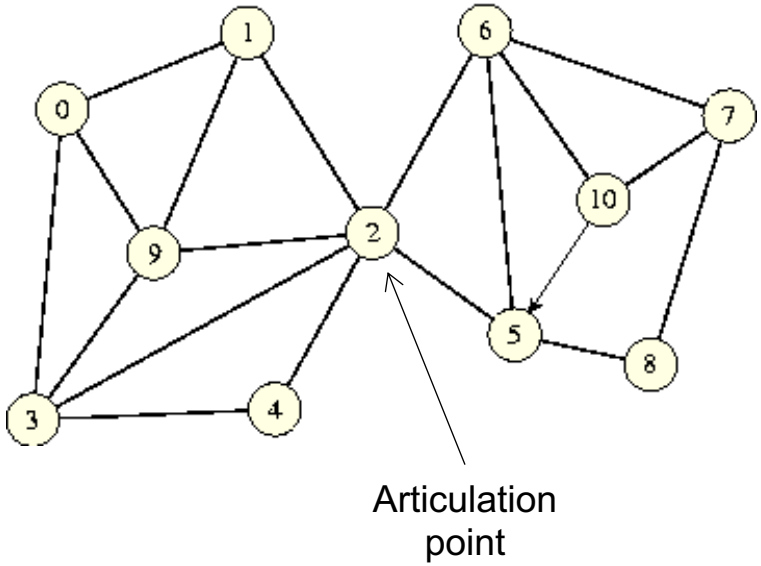
- ***Signed graph***
  - where the ties among nodes have signs assigned to them.



Other definitions

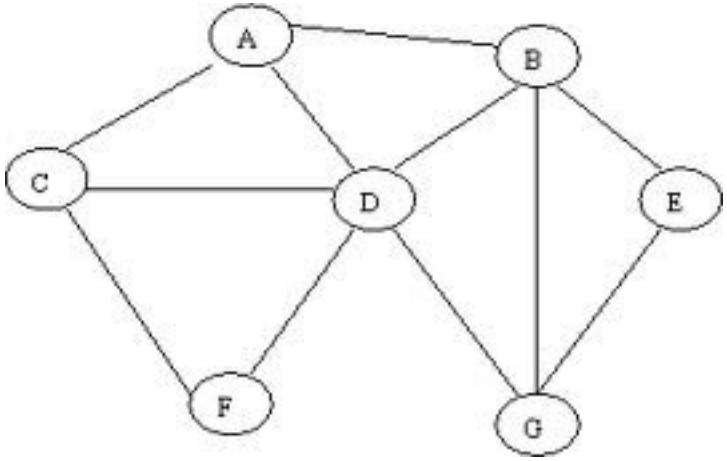
- ***Connected graph***

  when there is a path between every pair of vertices.



- ***Biconnected graphs***

  is a connected and "nonseparable" graph, meaning that if any vertex were to be removed, the graph will remain *connected*.

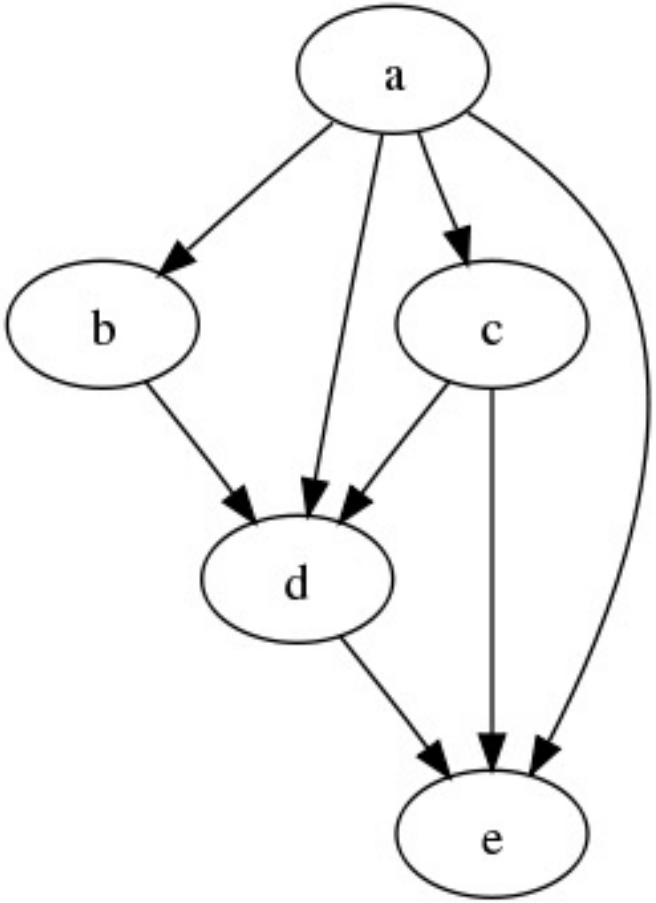


Other definitions

- ***Directed Acyclic Graph (DAG)***

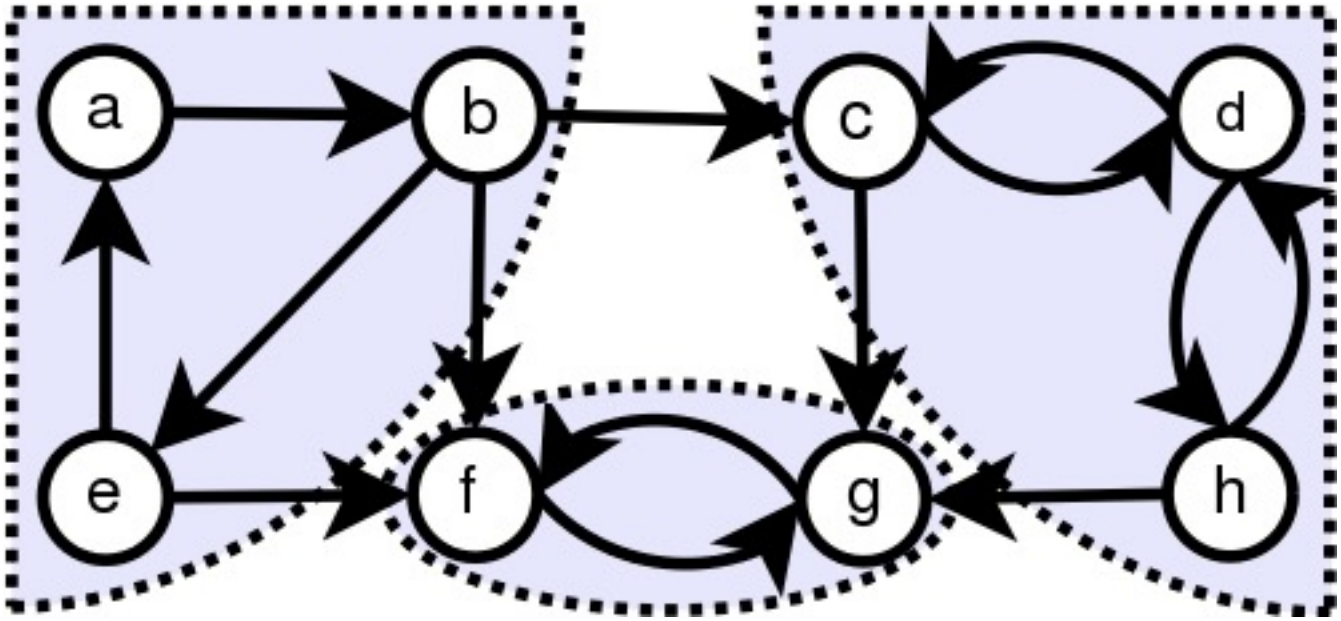
  Networks without cycles.

  (e.g.) papers' citation networks. When writing a paper, you can only cite another paper if it has been written, which means that all directed edges point backward in time.



Other definitions:
Strongly connected components

- It's a component of the graph in which there is a path in each direction between each pair of vertices of that component.



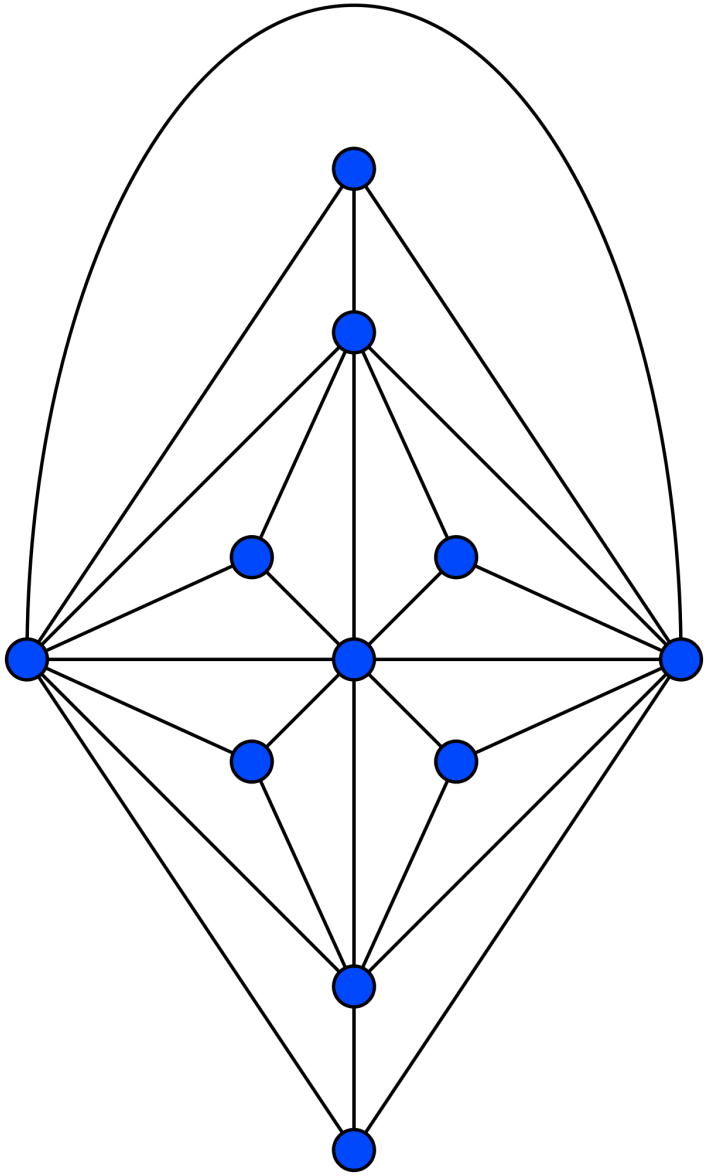
Other definitions

- ***Planar networks***

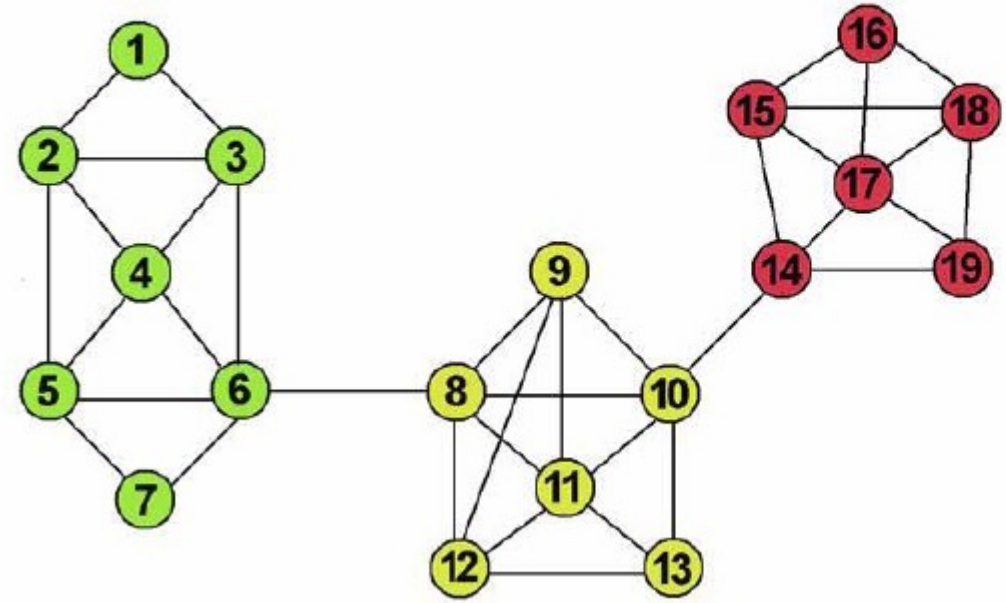
  Networks that can be drawn on a plane without having any edge cross.

  This is not very common in real world networks, but we find some (river networks, road networks (at least, most of the times), shared borders, etc.

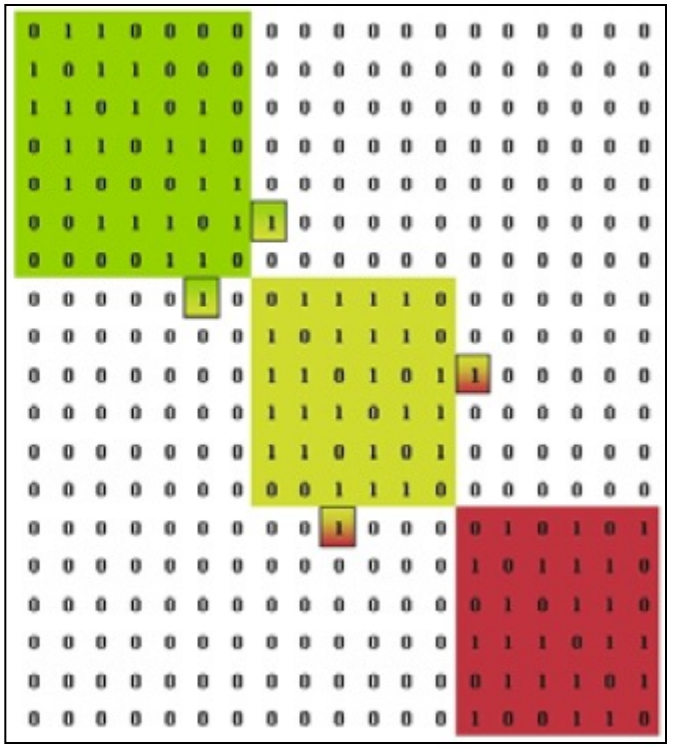
  They are mostly popular for theoretical problems (four colour theorem, etc) and other classic graph theory results.



How can we represent a network?

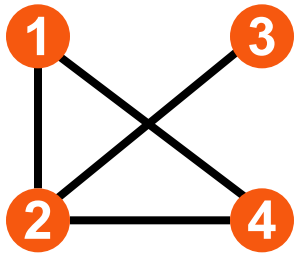


a_{ij} = Adjacency matrix



How can we represent a network?

undirected



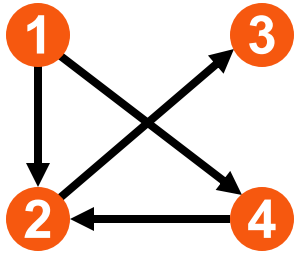
Adjacency lists

1	2	4	/	
2	1	3	4	/
3	2	/		
4	1	2	/	

Adjacency matrix

	1	2	3	4
1	0	1	0	1
2	1	0	1	1
3	0	1	0	0
4	1	1	0	0

directed

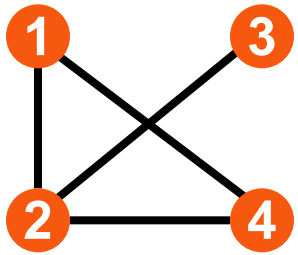


1	2	4	/
2	3	/	
3	/		
4	2	/	

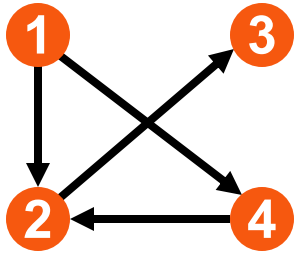
	1	2	3	4
1	0	1	0	1
2	0	0	1	0
3	0	0	0	0
4	0	1	0	0

How can we represent a network?

undirected



directed



Lists of edges

{1, 2}
{1, 4}
{2, 3}
{2, 4}

{1, 2}
{1, 4}
{2, 3}
{4, 2}

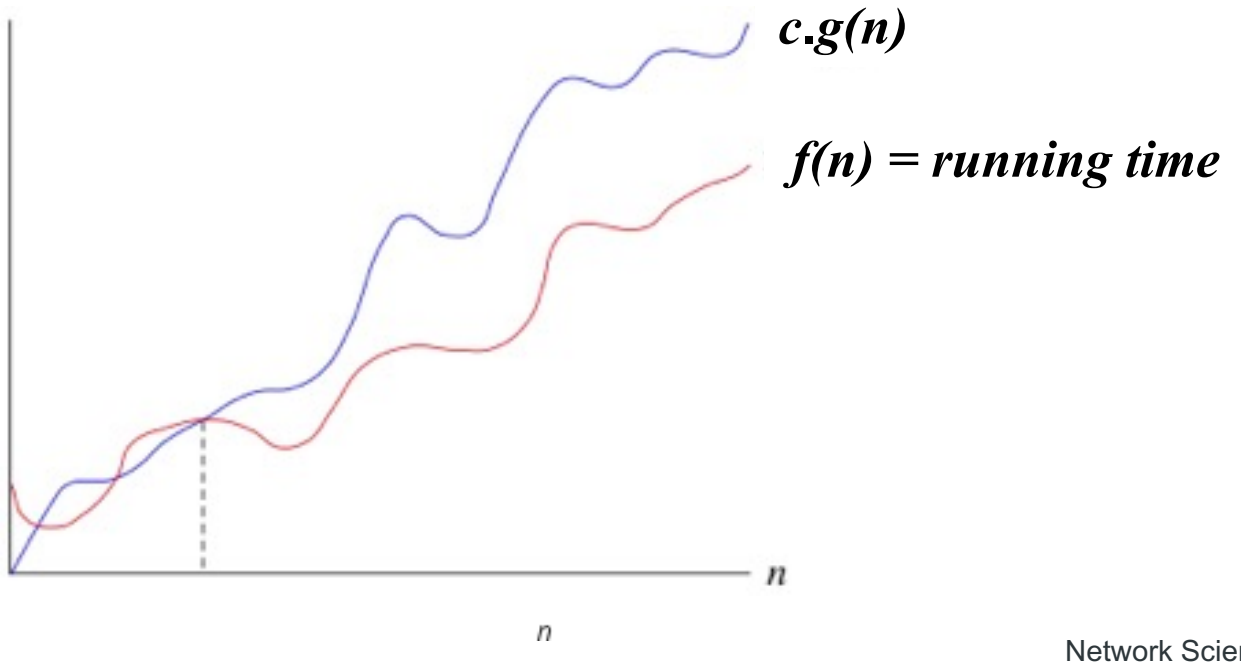
Alternative representations

- There may ways of representing a network
  - List of edges
  - Adjacency matrix
  - Adjacency lists
  - Compressed representations (we will return to this later in the course)

- Produce different performances when we try to access or modify their content.
- The “right” choice depends on the problem we aim at solving!
- **How can we formally evaluate the advantages and disadvantages of an algorithm or data structure?**

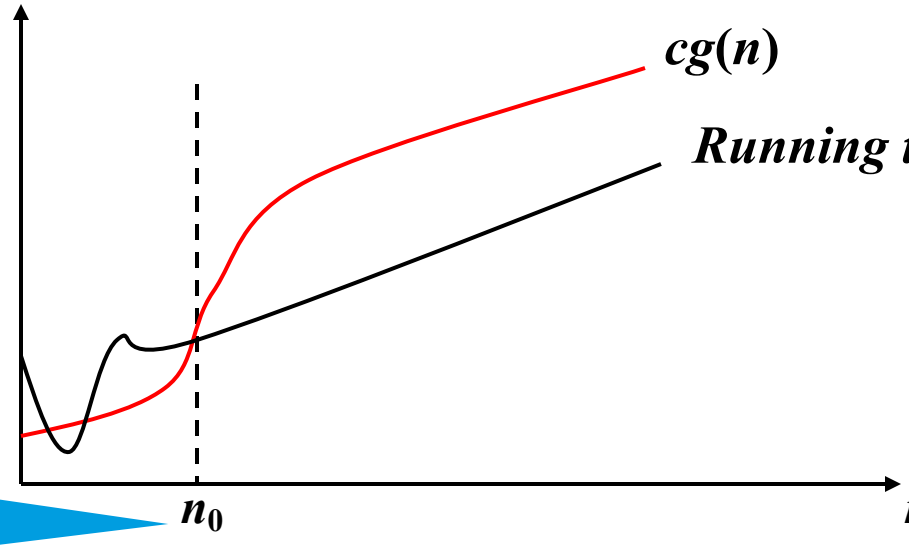
Asymptotic notation in a nutshell

- Imagine you want to determine how long an algorithm takes, in terms of the size of its input. [*time complexity*]
- …or how much memory it takes, once again, as a function of its input. [*space complexity*]



Big-*O* notation ⟶ the asymptotic upper bound

- We use big-O notation to ***asymptotically bound*** the growth of a running time or the space an algorithm of data structure takes.
- Example: if we say that the running time is $\mathbf{O}(g(n))$, then for large enough n, the running time is ***at most*** $c.g(n)$ **for some constant** c.
- Offers an upper bound of the **worst possible case**.



We only need a constant n_0 above which the condition is valid

Size of the input

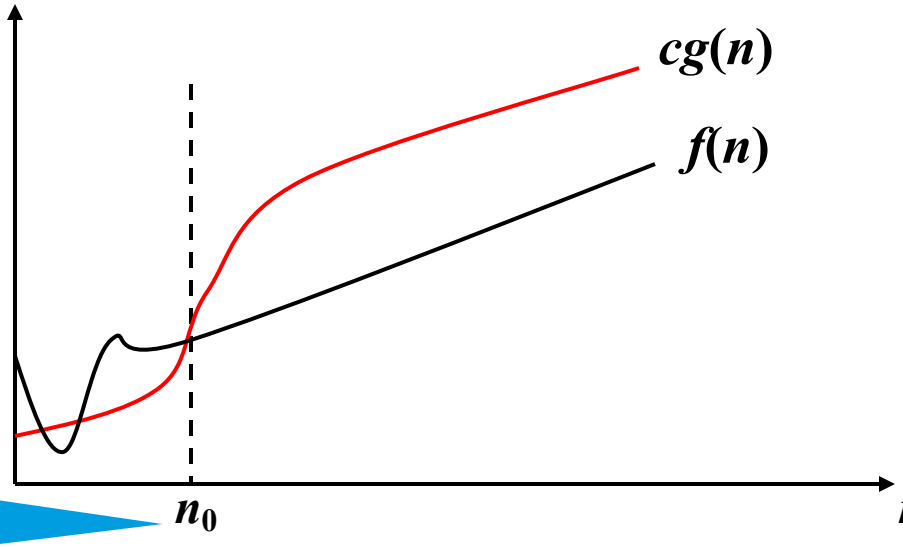
Asymptotic upper bound – the *Big-O*

- Notation O: Asymptotic ***upper*** bound
- **Assesses the worst possible case**

– $O(g(n)) =$

$\{ f(n)$: There are constants c and n_0, such that $0 \leq f(n) \leq cg(n)$, for $n \geq n_0 \}$

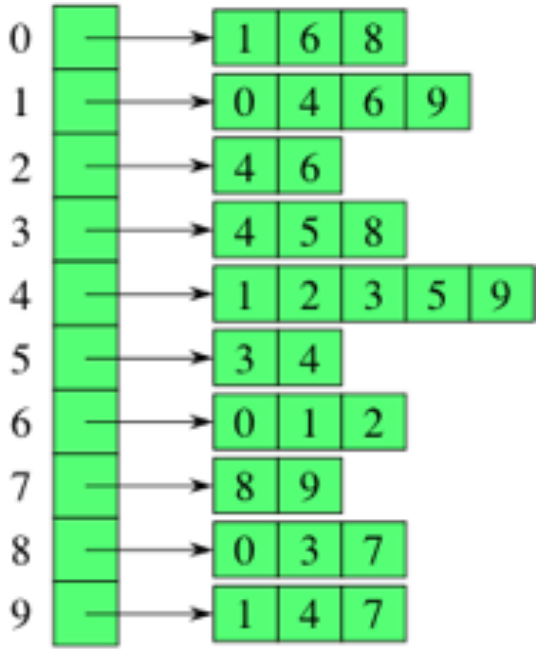
– $f(n) = O(g(n))$, means $f(n) \in O(g(n))$



We only need a constant n_0 above which $f(n) \leq cg(n)$

Size of the input

What's the worst possible scenario?
What's the best possible scenario?



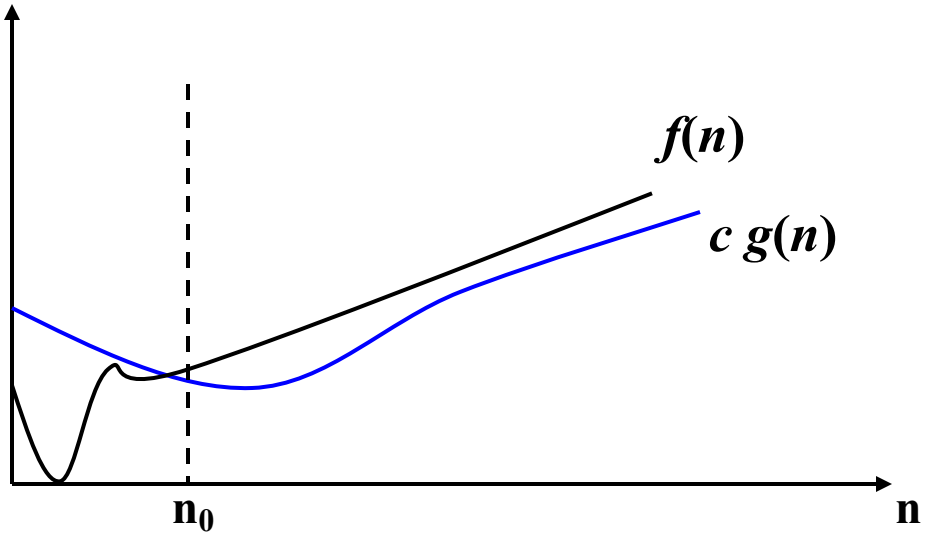
Asymptotic lower bound – the *Big*-Ω

- Notation Ω: Asymptotic ***lower*** bound
- **Assesses the best possible case**

– $\Omega(g(n)) =$

$\{ f(n)$: there are constants c and n_0, such that $0 \leq cg(n) \leq f(n)$, for $n \geq n_0 \}$

– $f(n) = \Omega(g(n))$, means $f(n) \in \Omega(g(n))$



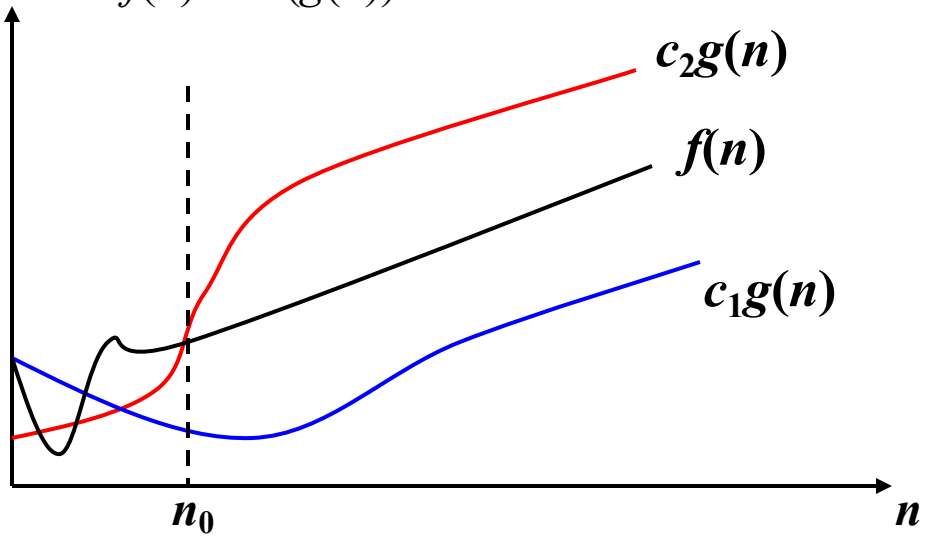
Asymptotic tight bound – the *Big*-Θ

- Notation Θ: Asymptotic ***tight*** bound
- **A function $f(n)$ is $\Theta(g(n))$ <u>if and only if</u> $f(n)$ is $O(g(n))$ and $\Omega(g(n))$**

– $\Theta(g(n)) =$

$\{ f(n)$: there are 3 positive constants c_1, c_2, and n_0, such that $0 \le c_1 g(n) \le f(n) \le c_2 g(n)$, for $n > n_0 \}$

– $f(n) = \Theta(g(n))$, means $f(n) \in \Theta(g(n))$



A few dumb questions... True or False?

- If an algorithm is $O(N^2)$ then it is also $O(N^3)$.

True (yet, useless!).

- If the execution time of an algorithm, in its worst case, scales with $3N^2$ then the algorithm is $O(N^2)$.

True (note that I can find another constant, say 6, which upper bounds $3N^2$) .

- The execution time of *G*, scales with $2N^3+N$ in the worst case and, in the best case, with N^3. Thus, G is $\Theta(N^3)$

True.

Let's get back to our networks

(with *N* nodes and *E* edges)

- The insertion of a new link in an adjacency matrix runs in O(**1**) time

- Removing a link from an adjacency list, in the worst case, is O(***N***).

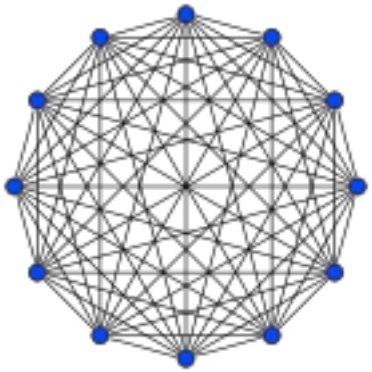
- The space occupied by an adjacency list is O(***N+E***)
- The space occupied by an adjacency matrix is O(N^2)

- How long does it take to check if two nodes are connected in each case?

O(1) (matrix) *and* ***O(N)*** (list)

Adjacency matrix - Advantages

- Suitable representation when
  - There's space available
  - Graphs are dense (*E* ➔ *N(N-1)/2*)



- Efficient insertion and removal of links.
- It's easy to avoid/control repeated edges.
- Easy to check if two nodes are connected.

Adjacency matrix - Disadvantages

- Large sparse graphs require memory space proportional to N^2
- In that case, just to setup the matrix requires N^2 steps, which may dominate the entire execution time of your algorithm.
- For sparse and large graphs, likely you won't have enough memory to build the matrix.

Adjacency lists - Advantages

- Initialization is proportional to N
- Requires space proportional to $N+E$
  - Suitable for sparse graphs
- Insertion of links is done efficiently ($O(1)$)

Adjacency lists - Disadvantages

- Checking repeated links
  - Require *loops over lists*, which may take a time proportional to the number of nodes N.
- Removal of links
  - Can take ~*N* operations.
- Not so great for…
  - Large graphs with repeated edges;
  - Operations with intensive removal of edges.

Performance

E = number of links ; N = number of nodes

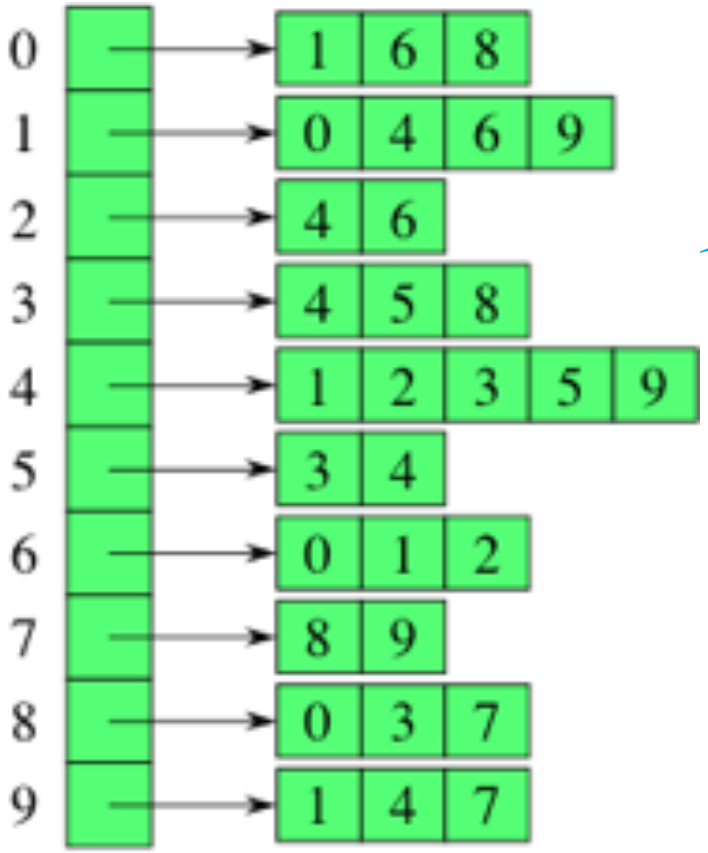
	Vector of edges	Adj. Matrix	Adj. Lists
Space	$O(E)$	$O(N^2)$	$O(N+E)$
Setup	$O(1)$	$O(N^2)$	$O(N)$
Copying a graph	$O(E)$	$O(N^2)$	$O(E)$
Destruction	$O(1)$	$O(N)$	$O(E)$
Link insertion	$O(1)$		
Finding a link	$O(E)$		
Removal of a link	$O(E)$		

Performance

E = number of links ; N = number of nodes

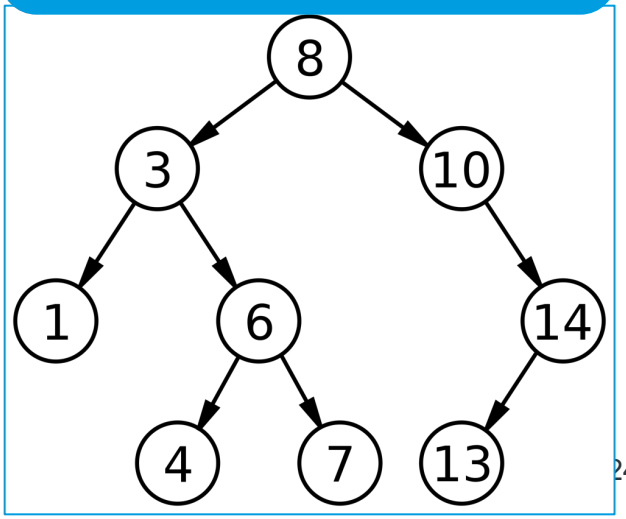
	Vector of edges	Adj. Matrix	Adj. Lists
Space	$O(E)$	$O(N^2)$	$O(N+E)$
Setup	$O(1)$	$O(N^2)$	$O(N)$
Copying a graph	$O(E)$	$O(N^2)$	$O(E)$
Destruction	$O(1)$	$O(N)$	$O(E)$
Link insertion	$O(1)$	$O(1)$	$O(1)$
Finding a link	$O(E)$	$O(1)$	$O(N)$
Removal of a link	$O(E)$	$O(1)$	$O(N)$

Possible improvements?

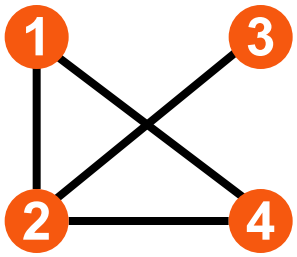


Searching for a partner may take a lot of time. Any solution?

Replace these linked lists (for instance) by a binary search tree.



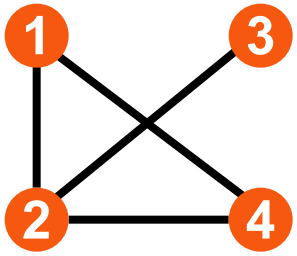
Representations with variables on edges
(e.g. weighted networks)



Adjacency matrix

	1	2	3	4
1	0	3	0	1
2	3	0	7	1
3	0	7	0	0
4	1	1	0	0

Use non-zero values to indicate weights or types of edges



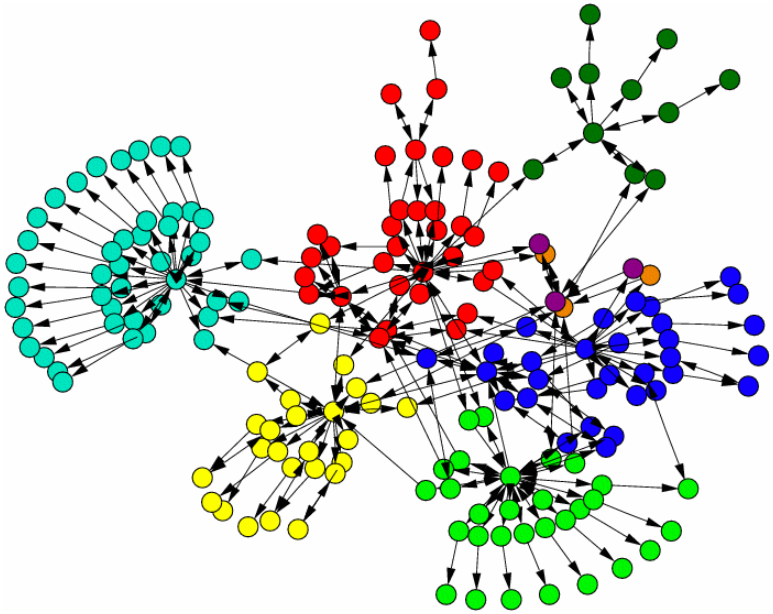
lists

1	2	4	/
2	1	3	4 /
3	2	/	
4	1	2	/

Replace each value by a structure that contains the details you want to store

How can we characterize a network?

Average distances, diameter, shortest paths, transitivity, clustering, etc, etc.



Among those, let us start by highlighting three:

- Degree distribution, *P(k)*
- Average path length, <L>
- Clustering Coefficient, C

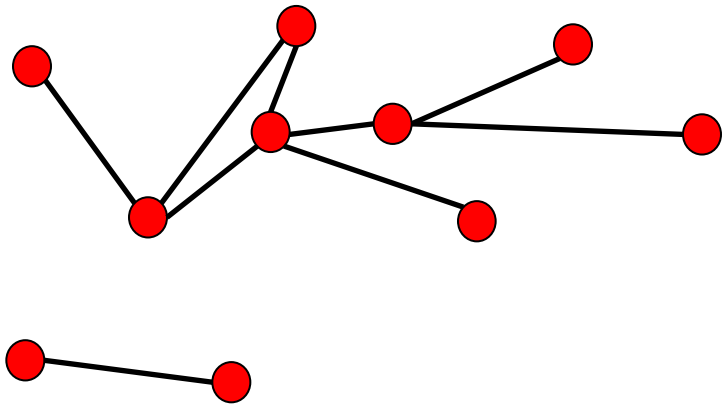
Degree distribution
(frequency distribution of nodes' degree)

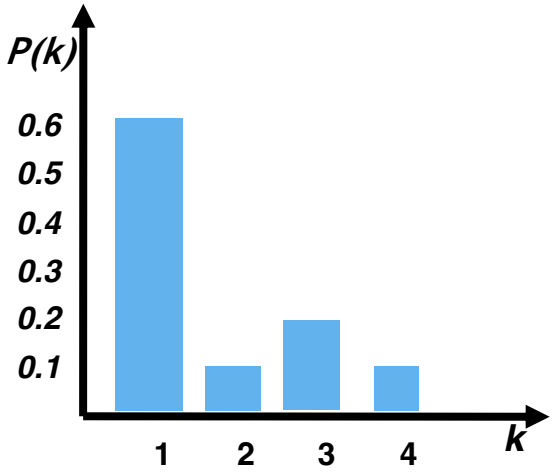
We have a sample of values $x_1, ..., x_N$ (in our case degrees of nodes)

Distribution of **x** (a.k.a. PDF*): probability that a randomly chosen value is **x**

$$P(x) = (\#\text{ values } x) / N$$

$$\Sigma_i P(x_i) = 1 \quad \text{always!}$$





*PDF: probabillity distribution function

Degree distribution

a_{ij} = Adjacency matrix

We already know what is the ***degree*** of a node i : $k_i = \sum_j a_{ij}$

degree distribution

P_k : probability of having a node of degree k ;

$$P_k = \frac{N_k}{N} = \frac{1}{N}\sum_i \delta(k_i - k)$$

average degree (z)

$$z \equiv \langle k \rangle = \sum_k k\, P_k = \frac{2E}{N} = \frac{1}{N}\sum_i k_i = \frac{1}{N}\sum_{ij} a_{ij}$$

Z is nothing but the first moment (n=1) of the degree dist.

$$\langle k^n \rangle = \sum_k k^n P_k$$

Degree distribution

discrete representation: P_k is the probability that a node has degree k.

continuum description: P(k) is the PDF of the degrees, where $\int_{k_1}^{k_2} P(k)\,dk$ represents the probability that a node's degree is between k_1 and k_2.

Normalization condition:

$$\sum_{k_{\min}}^{\infty} P_k = 1$$

$$\int_{k_{\min}}^{\infty} P(k)\,dk = 1$$

Where k_{min} is the minimal degree in the network.

Computing degree distributions

Degree of a node

It's handy, for instance if you have an Adj. List, to store the size of your list for each node, such that computing the degree of a particular vertex is simply looking at the position of an array, i.e., is $O(1)$.

Degree distribution

Once we have the degrees of the nodes, we just need to make an histogram. You may normalize it at the end of you wish.

Cumulative degree distribution

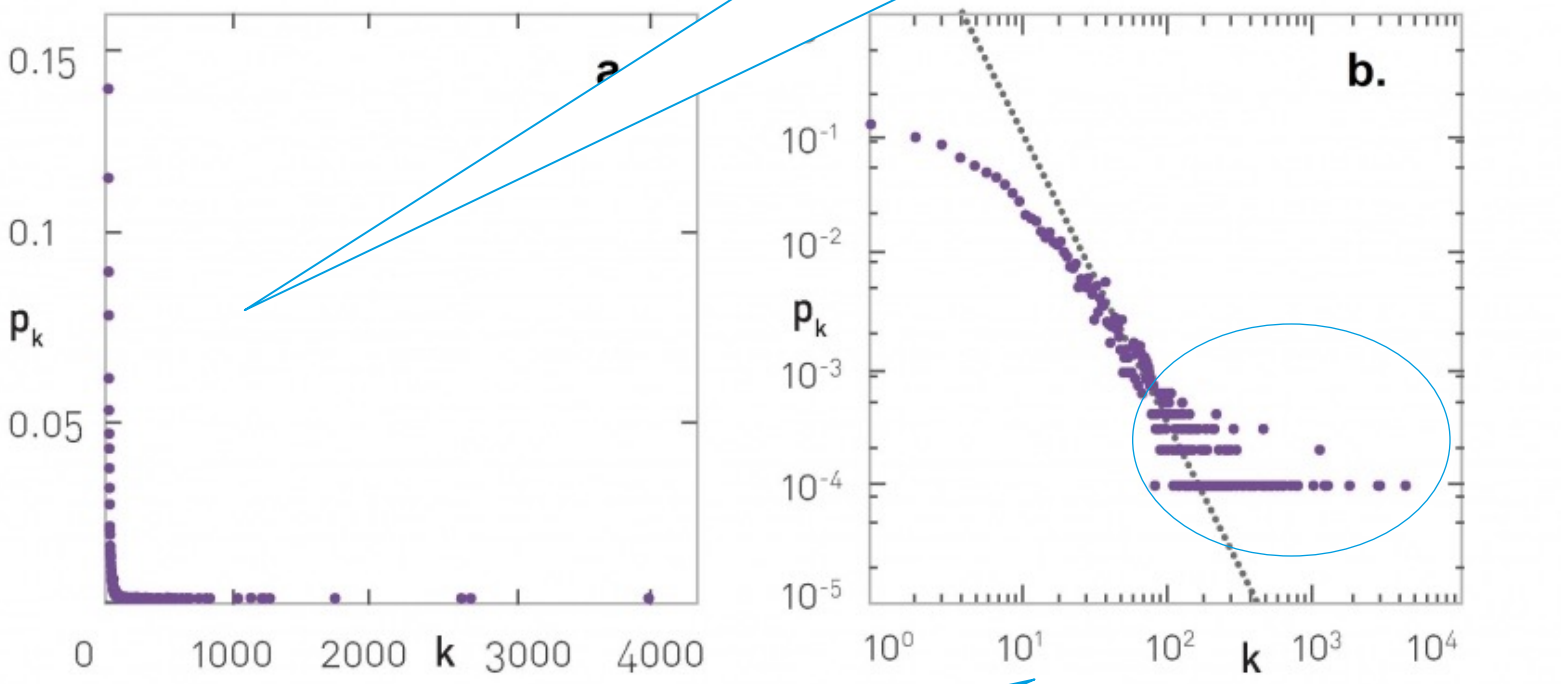
Once you have the degree dist, the cumulative degree dist is given by

$$P_k^{Cum} = \sum_{k'=k}^{\infty} P_{k'}$$

Plotting degree distributions

Typical power-law
In a linear scale

$$P_k \sim k^{-\gamma}$$



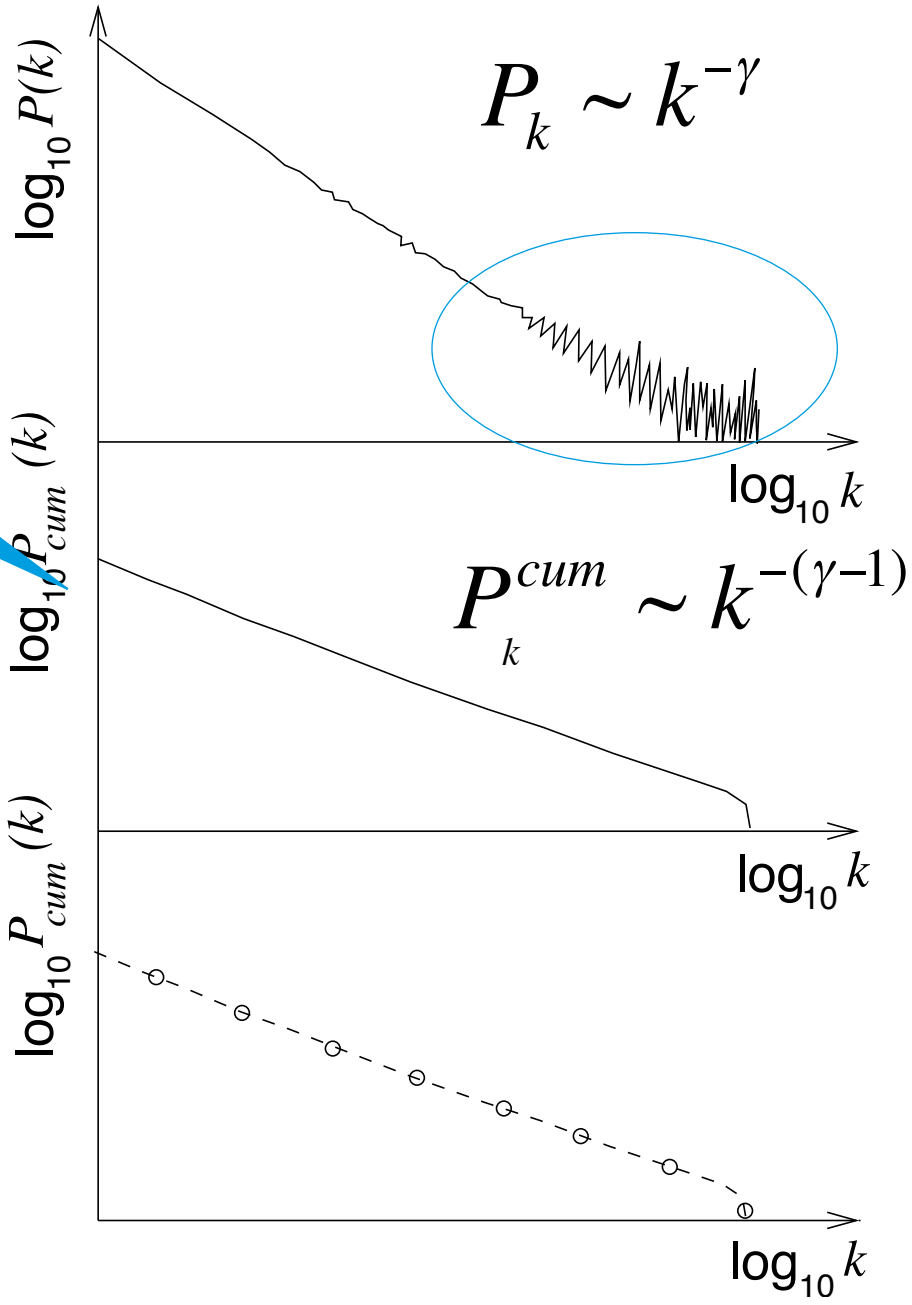
Same degree dist. in a log-log scale

Plotting degree distributions

By transforming the degree dist. into its cumulative, you get a clean power-law with a slope Y-1

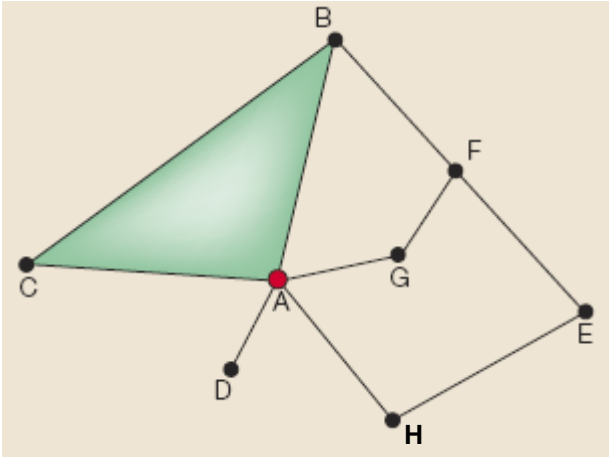
Your least-square fit will get get better if you average over equal size bins in the log scale.

Later we may return to this issue showing a more complex (yet, statistically correct) way of doing this procedure…



Average Path Length (APL), diameter, etc

shortest paths & average path length (APL) :



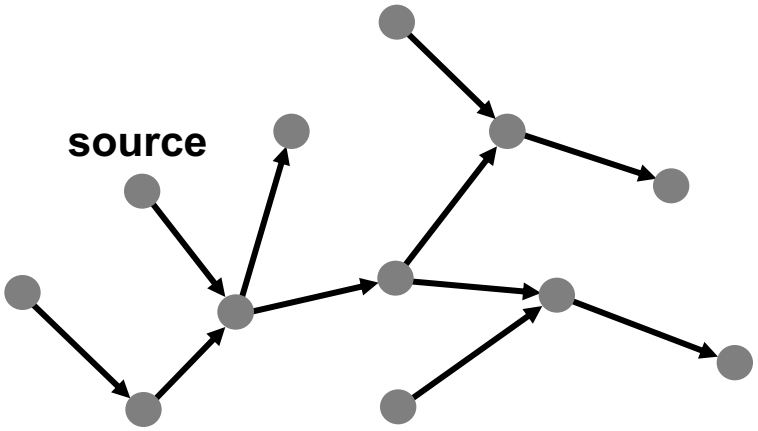
in the example above, there are various alternatives to join C & E; the shortest path L_{CE} is either [CBFE] or [CAHE], which have the same “length”, that is, the number of hops is 3.

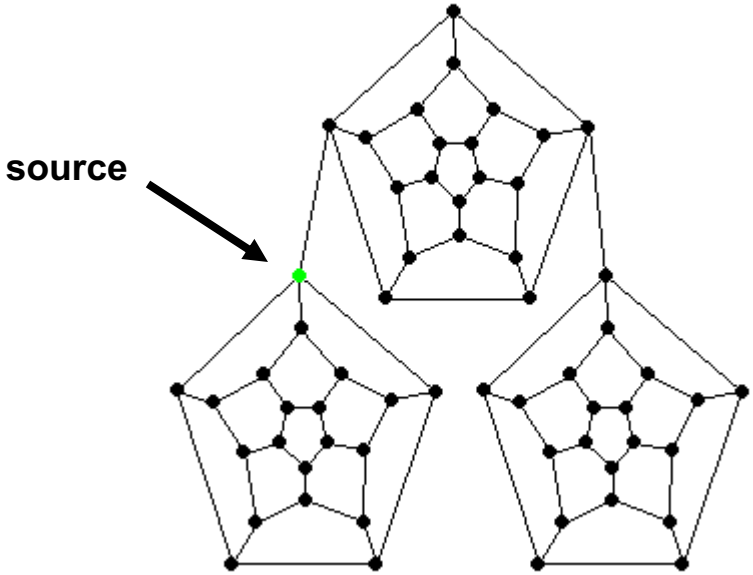
the average path length $<L>$ is the average over all shortest paths between all pairs of vertices of the graph :

$$\langle L \rangle = \frac{1}{N(N-1)} \sum_{ik} L_{ik}$$

Breadth-First Search (BFS)

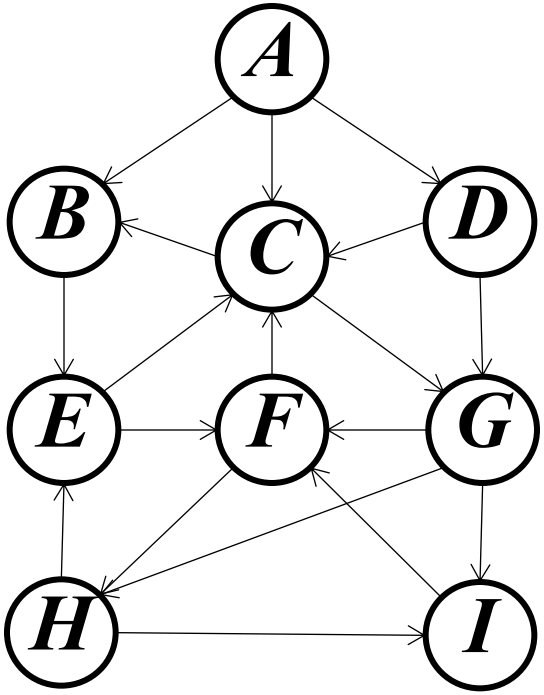
- BFS finds the shortest distance from a given starting node to every other node.
- Start by visiting all neighbours of the source. Then, visit all neighbours of the nodes you have just visited. And so on.





Breadth-First Search (BFS)

- Example



Adj. List

A: B, C, D
B: E
C: B, G
D: C, G
E: C, F
F: C, H
G: F, H, I
H: E, I
I: F

BFS depends on lists order

Sequence:	A	B	C	D	E	G	F	H	I
Distance:	0	1	1	1	2	2	3	3	3

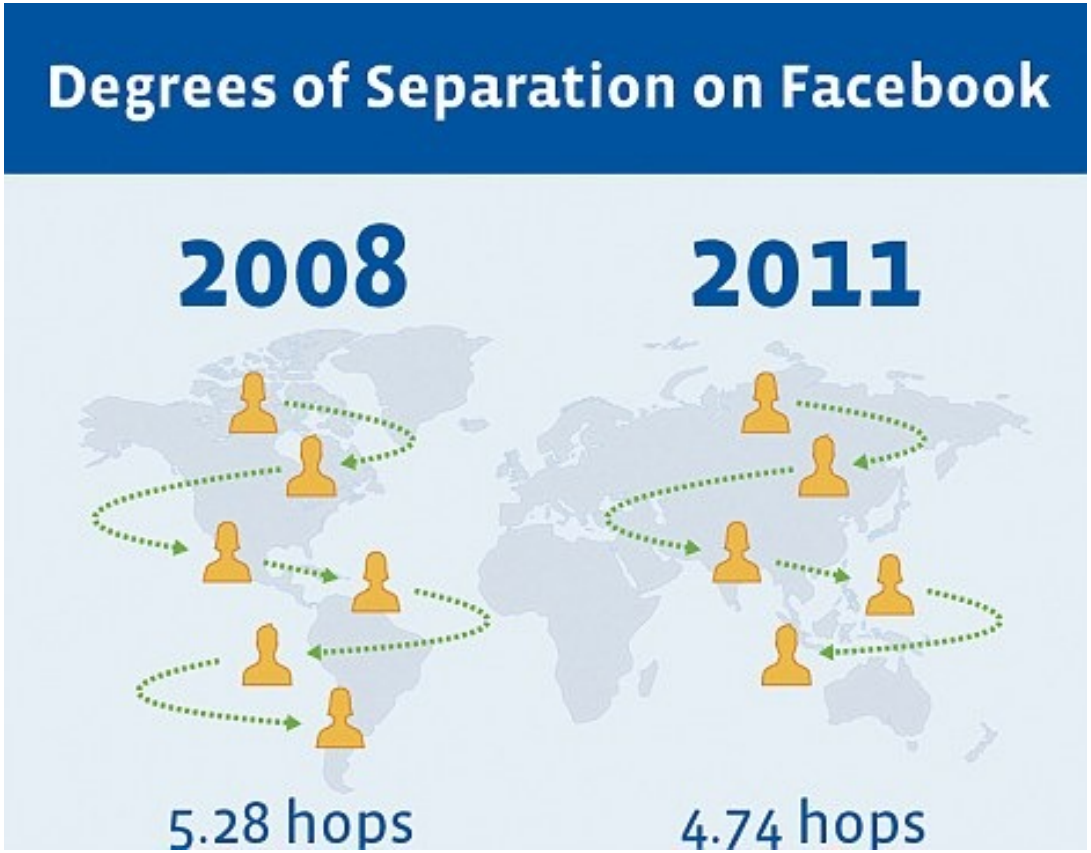
How long does it take?

- For a given starting node, it takes O(*N*+*E*) (w/ linked lists).
- Since most of our networks are sparse, we get ~O(*N*).

- If you want to compute the average path length (APL, i.e., all distances) and naively apply several BFS's you would get $O(N^2)$ for sparse graphs.

- Yet, sometimes this is not enough…☺

Playing with large networks…



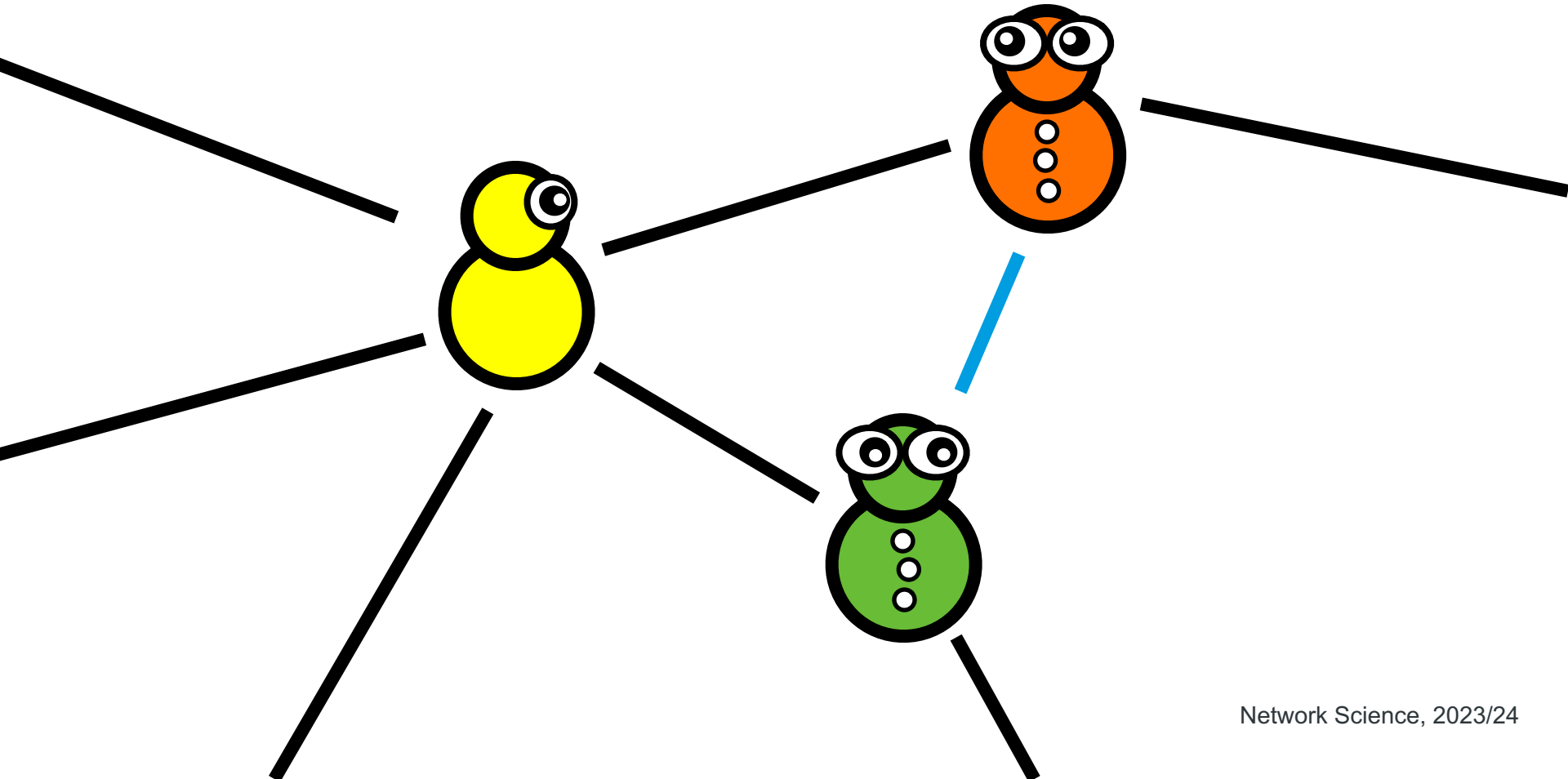
http://law.di.unimi.it/webdata/fb-current

Facebook: 1.6×10^9 users!!!

Clustering coefficient or transitivity

mind your friends' friends

People tend to have friends who are also friends with each other.

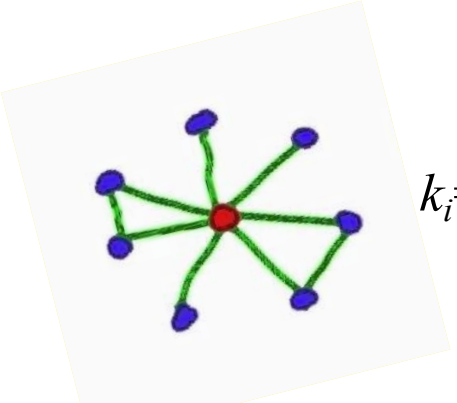


Clustering coefficient

mind your friends' friends

People tend to have friends who are also friends with each other.
How do we measure this?

$$C_i = \frac{\#\text{ edges among neigs.}}{\max \#\text{ edges among neigs.}} = \frac{e_i}{k_i(k_i-1)/2}$$



k_i=7, e_i=2, thus $\quad C_i = \frac{2}{7(7-1)/2} = 0.095$

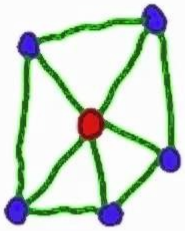
The network clustering coefficient is the average of the clustering coefficients for all nodes

$$\langle C \rangle = \frac{1}{N}\sum_i C_i$$

Challenge 1

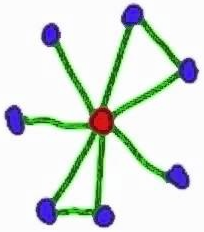
$$C_i = \frac{\#\text{ edges among neigs.}}{\max\ \#\text{ edges among neigs.}} = \frac{e_i}{k_i(k_i-1)/2}$$

A



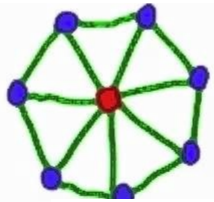
$$C_i = \frac{5}{5(5-1)/2} = 0.5$$

B



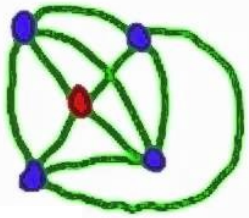
$$C_i = \frac{2}{7(7-1)/2} = 0.09$$

C



$$C_i = \frac{7}{7(7-1)/2} = 0.3$$

D



$$C_i = \frac{6}{4(4-1)/2} = 1.0$$

Order by clustering coefficient of the red node of each graph (lowest to highest):

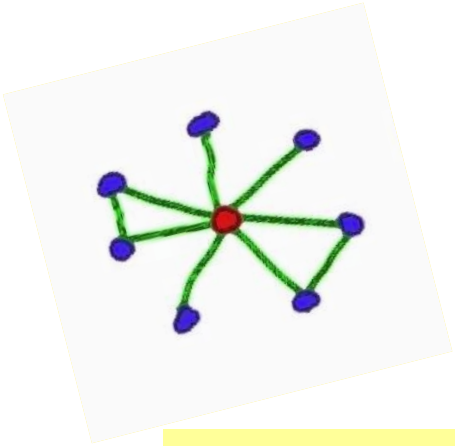
B, C, A, D

Clustering coefficient

mind your friends' friends

Computing the clustering coefficient of a node

$$C_i = \frac{\#\text{ edges among neigs.}}{\max \#\text{ edges among neigs.}} = \frac{e_i}{k_i(k_i-1)/2}$$



$$\langle C \rangle = \frac{1}{N}\sum_i C_i$$

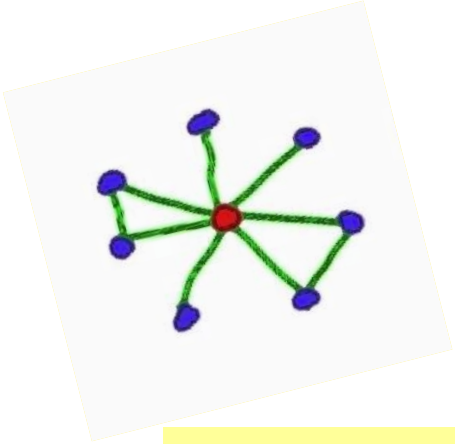
Loop over every pair of neighbors, and count how many are connected. We only need to consider each pair (j, i) once, such that j<i.

Clustering coefficient

mind your friends' friends

Computing the clustering coefficient of a node

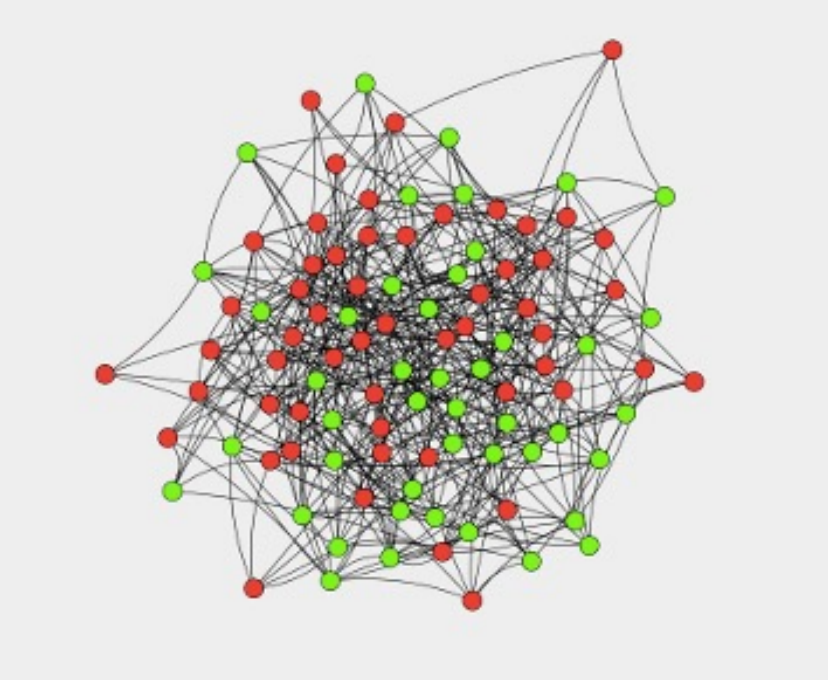
$$C_i = \frac{\#\text{ edges among neigs.}}{\max \#\text{ edges among neigs.}} = \frac{e_i}{k_i(k_i-1)/2}$$



$$\langle C \rangle = \frac{1}{N}\sum_i C_i$$

The clustering coefficient of a network is given by the average over all clustering coefficients

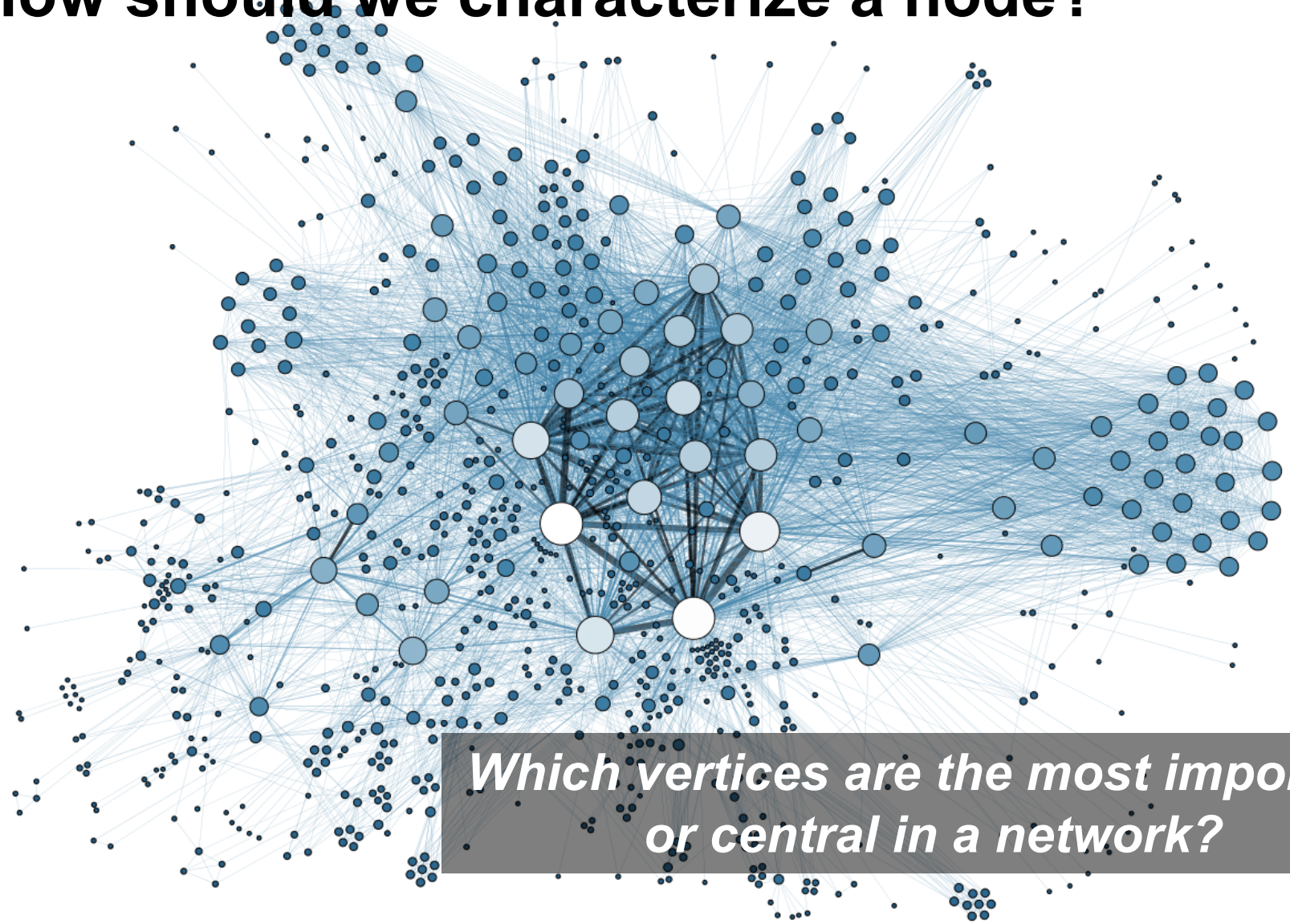
We are starting to have an idea on how to characterize a network through global measures ☺



- Degree distribution, *P(k)*
- Average degree, <k>
- Average path length, <L>
- Clustering Coefficient, <C>

(we will return to these ideas later)

How should we characterize a node?



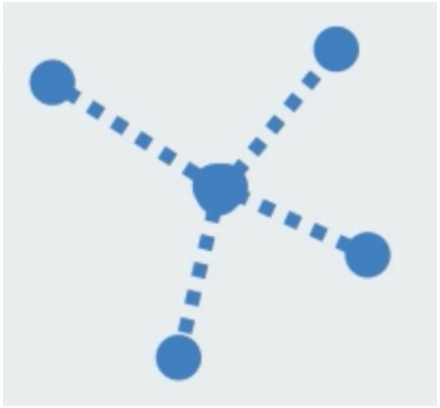
Which vertices are the most important or central in a network?

Centrality or *Importance* should depend on the context

4

criteria

Centrality or *Importance* should depend on the context



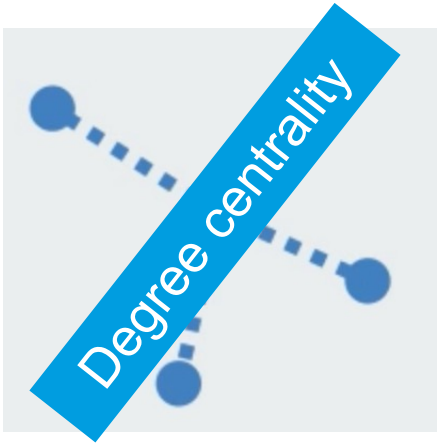
How popular you are

How prestigious your friends are

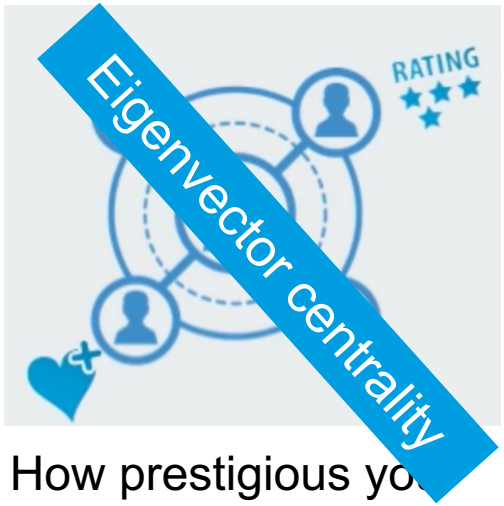
How close you are

How do you influence the flow of information, or bridge different parts of the network

Centrality or *Importance* should depend on the context



How popular you are

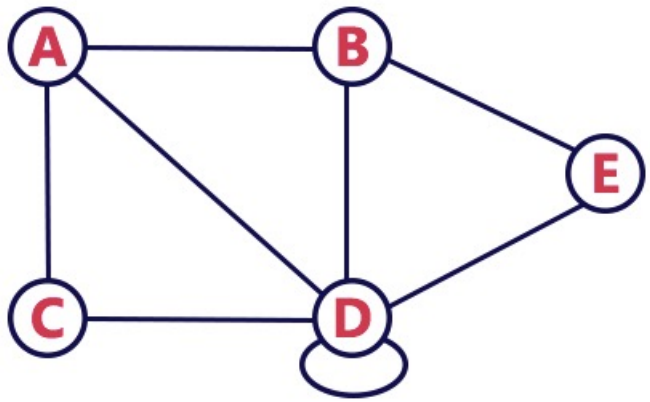


How prestigious yo
friends are

How close you are

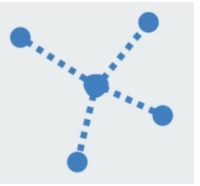
How do you influence the flow
rmation,
or bridge different parts of the
network

Again, let's consider an adjacency matrix a_{ij}



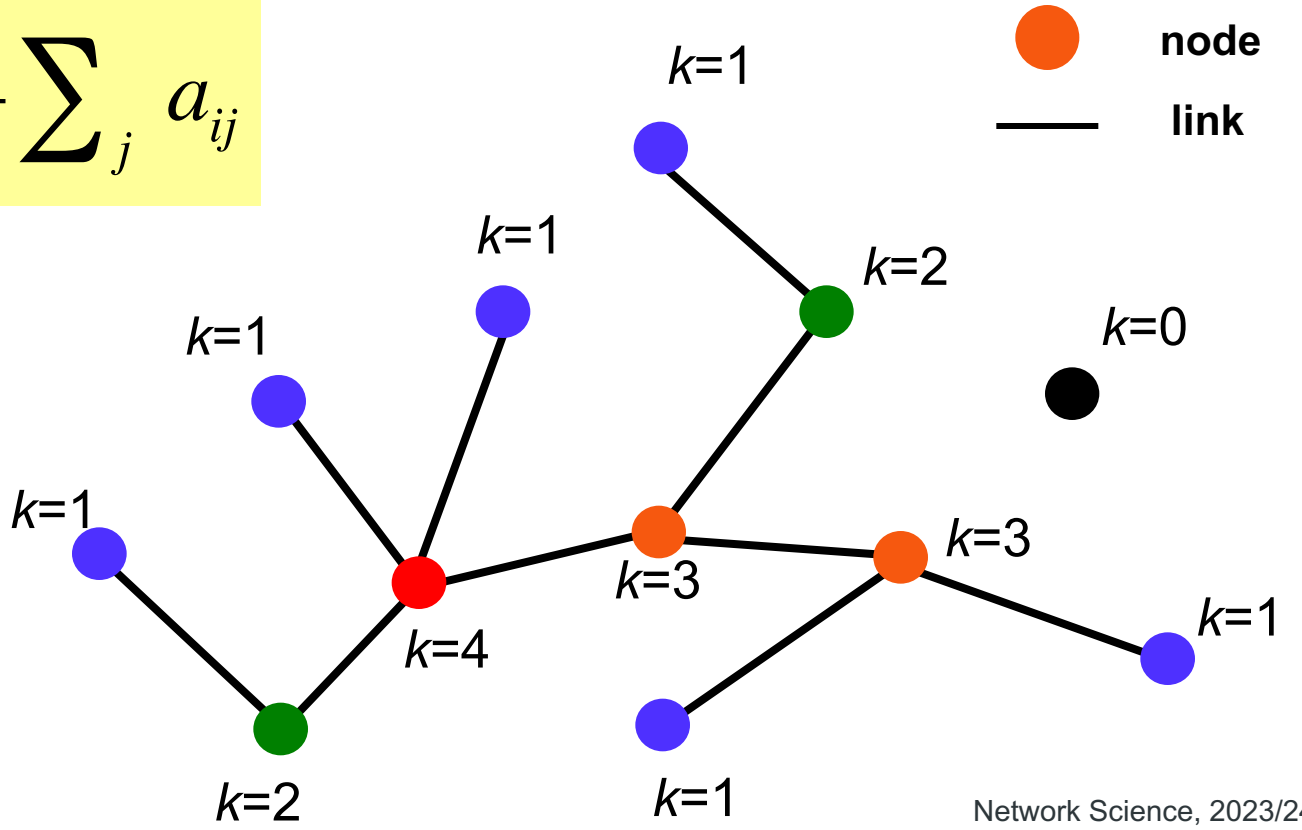
	A	B	C	D	E
A	0	1	1	1	0
B	1	0	0	1	1
C	1	0	0	1	0
D	1	1	1	1	1
E	0	1	0	1	0

Degree centrality (C_D)



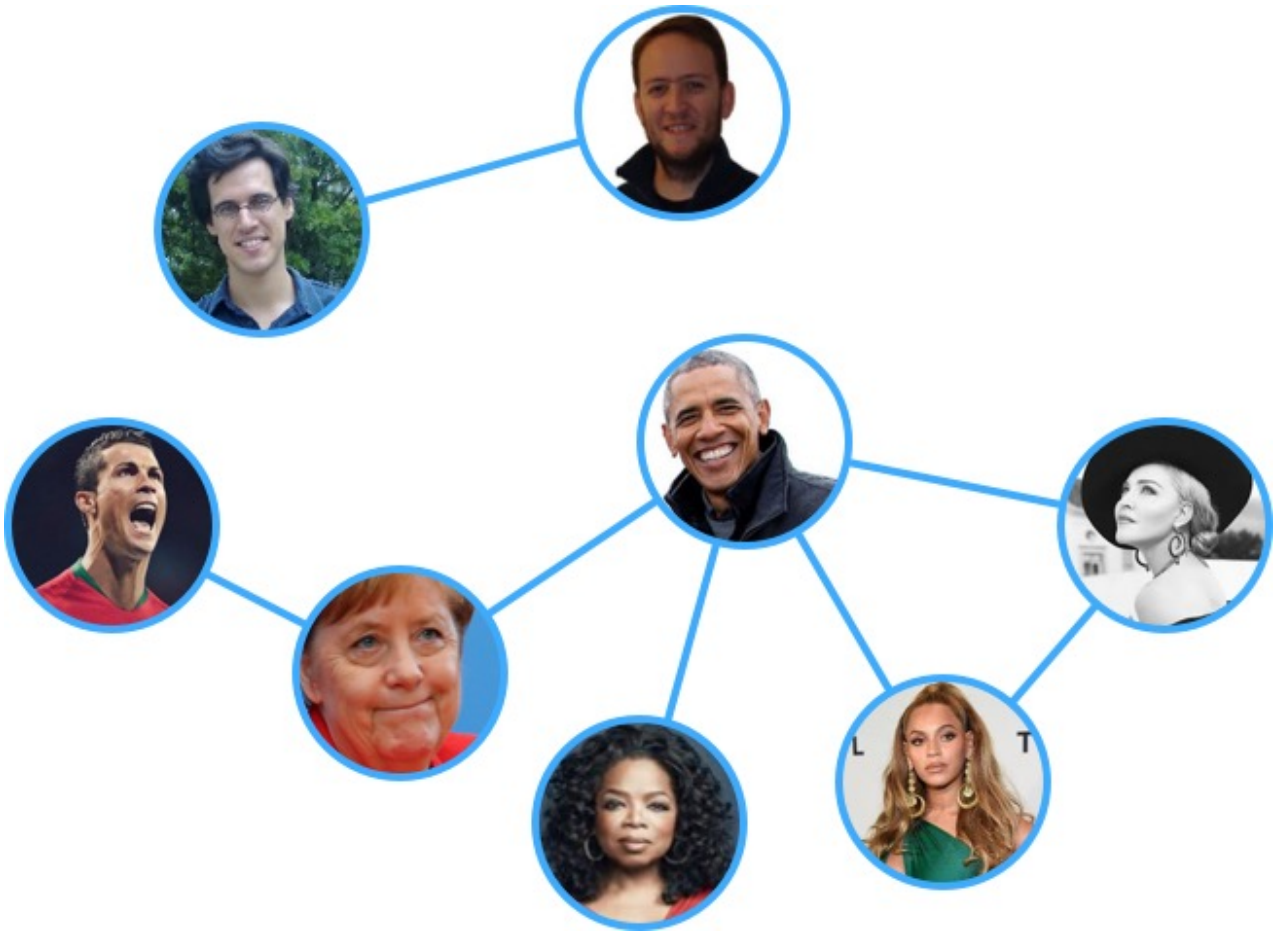
The most obvious centrality measure is the degree (often normalized by *N-1*).

$$C_D(i) = \frac{1}{N-1} \sum_j a_{ij}$$



my friends are better than yours

- A natural extension is to consider the importance of the nodes each node is connected with:
  Not all neighbors are equivalent.



my friends are better than yours

- A natural extension is to consider the importance of the nodes each node is connected with:
  Not all neighbors are equivalent.

- Thus, the vertex importance increases with neighbors that are themselves important.

- Instead of awarding a “point” for each neighbor, ***Eigenvector centrality*** **gives each node a score proportional to the sum of the scores of its neighbors**.

Philip Bonacich, 1972

Eigenvector centrality

My friends are better than yours!

- How can we formalize this (recursively)? The centrality x_i of node *i* can be written as

$$x(i) = \sum_{j \in Neigs(i)} x_j = \sum_j a_{ij} x_j$$

- But I don't know the centrality of the neighbors of *i* ☹...
- I may compute all at the same time, considering the vector **x** of all centralities (**A**=adj. matrix) …

$$\mathbf{x}(t=1) = \mathbf{A}\mathbf{x}(t=0)$$

- Now let's iterate many times (say t times) to get a good estimate (assuming, e.g., that x_j=1 for t=0, and all *j*).

Eigenvector centrality

My friends are better than yours!

If you keep going, it would be handy that for t➔∞ it converges, such that

$$\mathbf{x} = \mathbf{A}\mathbf{x}$$

Since this won't happen, we need a dumping factor λ

$$\mathbf{x} = \frac{1}{\lambda}\mathbf{A}\mathbf{x} \quad \Rightarrow \quad \lambda\mathbf{x} = \mathbf{A}\mathbf{x}$$

Reaching to a eigenvector problem.

x is a non-negative eigenvector
➔λ will be the highest eigenvalue

Eigenvector centrality

My friends are better than yours!

- ***Linear Algebra offers some help here***:

Since that, if we demand that all the entries in the eigenvector be non negative then (by the Perron–Frobenius theorem) only the greatest eigenvalue ensures the desired centrality measure.

In other words, **x** is the eigenvector of **A** associated with its largest eigenvalue λ. Often it is also common to normalize the values of **x** such that

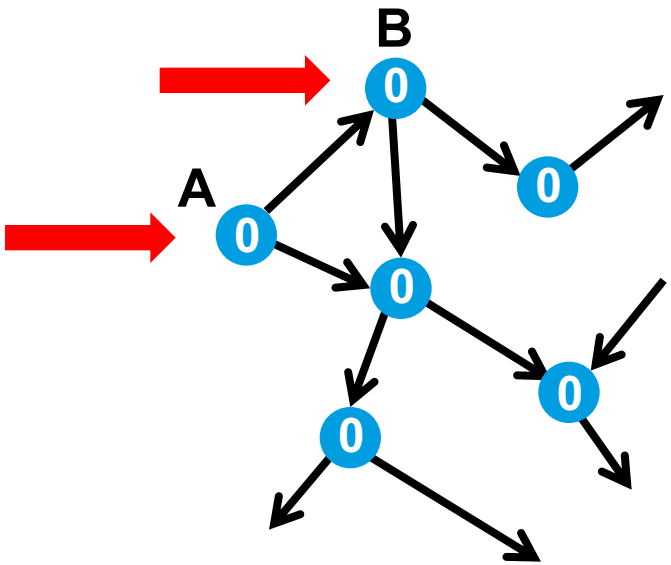
$$\sum_j x_j = 1$$

See, e.g., the mathematics of networks by M. E. J. Newman

Eigenvector centrality in directed graphs

My friends are better than yours!

- ***Eigenvector centrality has some issues with directed graphs…*** Normally it is defined based on the neighbors pointing at you (in-links)



- ***A*** will have centrality 0… OK. What about ***B***? Since ***A*** has x=0, ***B*** will also have centrality equal to 0.
- NOTE: Formally, only nodes belonging to ***strongly connected components*** will escape this issue.

Katz centrality

My friends are better than yours!

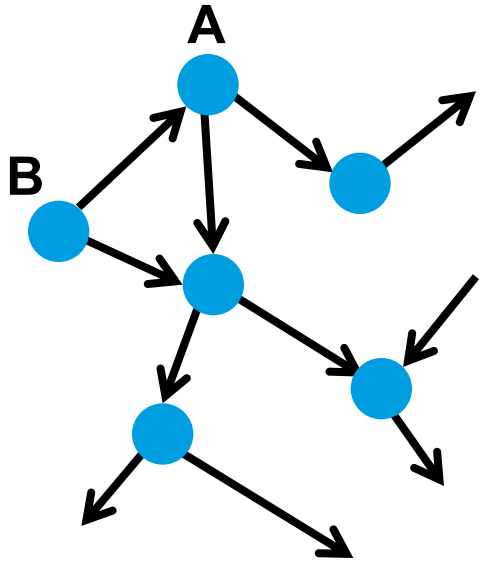
- ***Tries to solve this problem by adding a constant β…***

Eigenvector centrality

$$x(i) = \frac{1}{\lambda} \sum_j a_{ij} x_j$$

Katz centrality

$$x(i) = \alpha \sum_j a_{ij} x_j + \beta$$



- Even vertices with in-degree = 0 (node A) still get centrality β... The same happens to those that receive a link from these nodes (node B).

- The actual value of β can be shown to be unimportant (usually β=1).
- It has a free variable α which needs to be set, balancing the power of the free term. Also, α cannot be too large ($\alpha < 1/\lambda$).

Katz centrality

My friends are better than yours!

- *Katz centrality also has some problems…*
  Should all in-links be considered as equivalent?

If a vertex with high Katz has edges pointing to one-million others, it will give all one million of them a high centrality. This is not always appropriate.

Imagine that G-Scholar (GS, node with high centrality) has a link to my webpage. Thus, this would mean that my webpage should also have a high centrality? ***No!!!***

Solution: ***PageRank***

The contribution that GS offers to my webpage is divided by GS's out-degree.

Page Rank

My friends are better than yours!

- Variation of Katz in which the centrality that I receive in my webpage i from a node j is divided by j's *out-degree*.

Katz centrality

$$x(i) = \alpha \sum_j a_{ij} x_j + \beta$$

PageRank

$$x(i) = \alpha \sum_j a_{ij} \frac{x_j}{k_j^{out}} + \beta$$

Note 1: Vertices with 0 out-degree are artificially set with $k^{out} = 1$

Note 2: It is often set $\beta = 1$

Note 3: As with Katz, it has α as a free parameter. *Google* search engine is said to use a value of $\alpha = 0.85$, likely based on experimentation to find what works well…

Page Rank

My friends are better than yours!

- Variation of Katz: If a page i has an importance x_i and k_i^{out} out-links, each link gets x_i / k_i^{out}

Katz centrality

$$x(i) = \alpha \sum_j a_{ij} x_j + \beta$$

PageRank

$$x(i) = \alpha \sum_j a_{ij} \frac{x_j}{k_j^{out}} + \beta$$

Note 1: Vertices with 0 out-degree are artificially set with $k^{out} = 1$

Note 2: It is often set $\beta = 1$

Note 3: As with Katz, it has α as a free parameter. *Google* search engine is said to use a value of $\alpha = 0.85$, likely based on experimentation to find what works well…

Closeness centrality (C_C)

How close an individual is to all other nodes of the network?

Closeness centrality (C_C)

How close you are from everyone else?

How close an individual is to all other nodes of the network?

The average distance is given by $$l(i) = \frac{1}{N-1}\sum_{j(\neq i)} d_{ij}$$

Yet, if we simply pick the average dist. we would get low values for high centrality, which looks strange. So let us "invert" the idea to

$$C_C(i) = \frac{1}{l(i)} = \frac{N-1}{\sum_{j(\neq i)} d_{ij}} \approx \frac{N}{\sum_{j} d_{ij}}$$

Only works for connected components

$d_{ij} = \infty$

An alternative: Harmonic centrality (C_H)

How close you are from everyone else?

How close an individual is to all other nodes of the network?

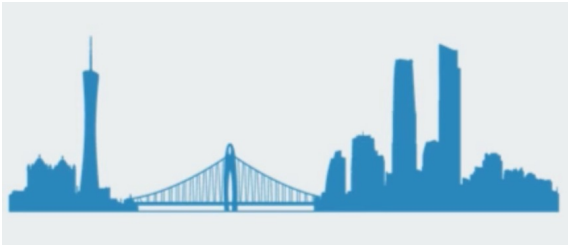
To avoid the problem of infinite distances one can use the harmonic mean distance between vertices
(i.e., the average of the inverse distances)

$$C_H(i) = \frac{1}{N-1} \sum_{j(\neq i)} \frac{1}{d_{ij}}$$

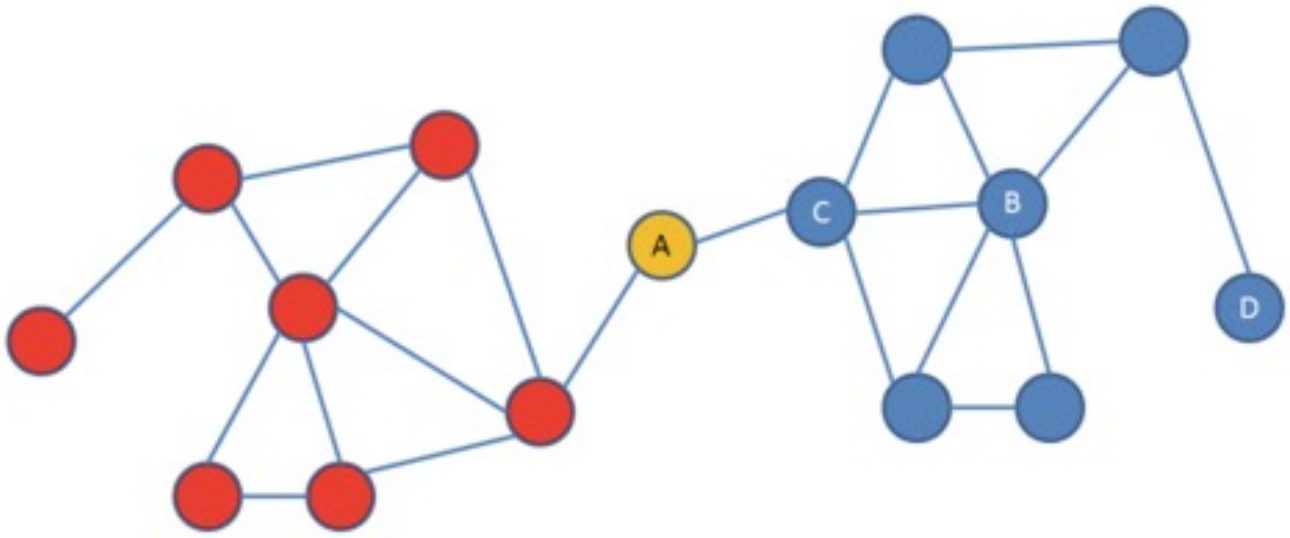
- It nicely allows $d_{ij} = \infty$.
- Gives more weight to vertices close to *i*, i.e., those that are far way do not matter how exactly they distance; their contributions are close to zero.

Betweenness centrality (C_B)

You're on everyone's way!

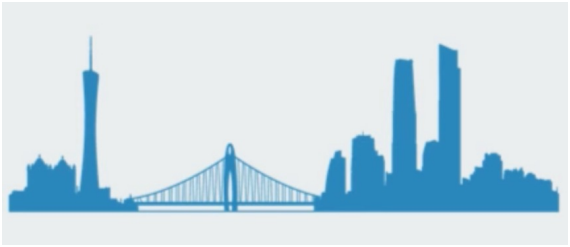


Measures the extent to which a vertex lies on paths between other vertices.



Betweenness centrality (C_B)

You're on everyone's way!



Number of shortest paths from node s to t that pass through i

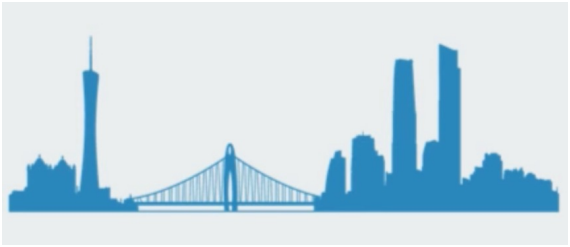
$$C_B(i) = \sum_{s \neq t \neq i} \frac{\sigma_{st}(i)}{\sigma_{st}}$$

Scales with the number of pairs

Total number of shortest paths from node s to t

Betweenness centrality (C_B)

You're on everyone's way!



Number of shortest paths from node s to t that pass through i

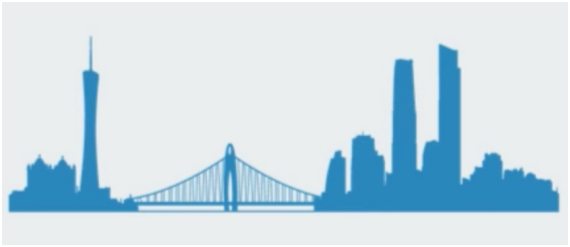
$$C_B(i) = \frac{1}{N^2} \sum_{s \neq t \neq i} \frac{\sigma_{st}(i)}{\sigma_{st}}$$

Total number of shortest paths from node s to t

One natural choice is to divide by the total number of ordered pairs which ~N^2

Betweenness centrality (C_B)

You're on everyone's way!



- Betweenness is a measure of the influence a node has over the spread of information through the network.
- By counting only shortest paths, however, the conventional definition implicitly assumes that information spreads only along those shortest paths.

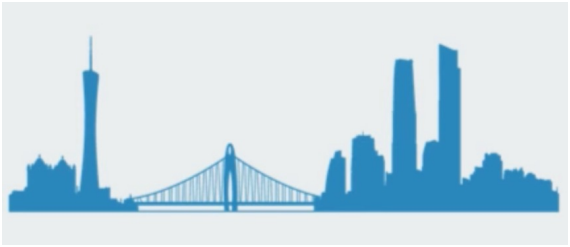
- **Alternative**: Random walks

Include contributions from all paths between nodes, not just the shortest by counting how often a node is traversed by a random walker.

Newman, M. E. (2005). A measure of betweenness centrality based on random walks. Social networks, 27(1), 39-54.

Betweenness centrality (C_B)

You're on everyone's way!



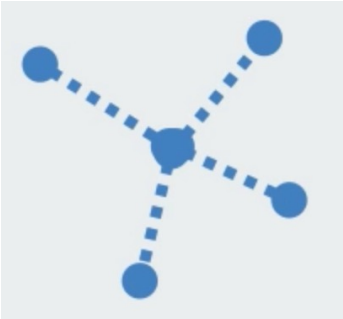
- Betweenness is a measure of the influence a node has over the spread of information through the network.
- By counting only shortest paths, however, the conventional definition implicitly assumes that information spreads only along those shortest paths.

- **Alternative**: Random walks

There's also a close connection between random walks and the PageRank: PageRank is a way to organize random walk of various lengths (F Chung & W Zhao, *Fete of combinatorics and computer science*, 2010)

Newman, M. E. (2005). A measure of betweenness centrality based on random walks. Social networks, 27(1), 39-54.

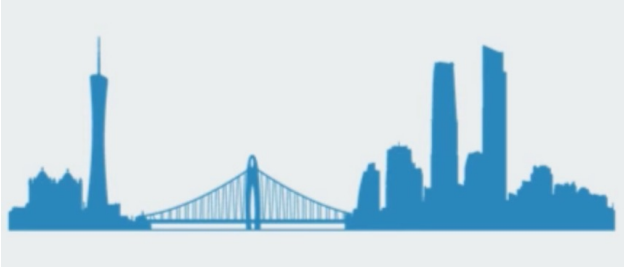
different metrics, different concepts.



Degree centrality

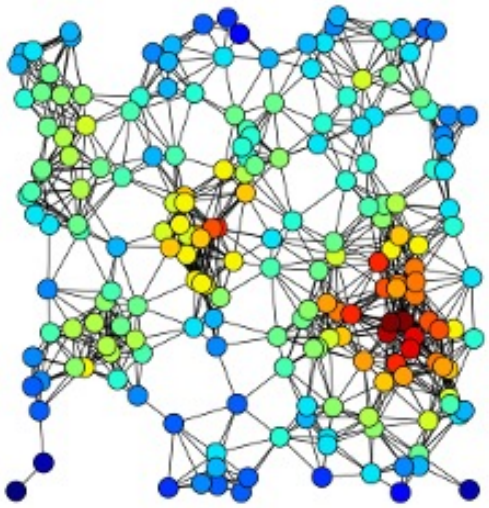
Eigenvector, Katz, PageRank centralities

Closeness Centrality & Harmonic Centrality



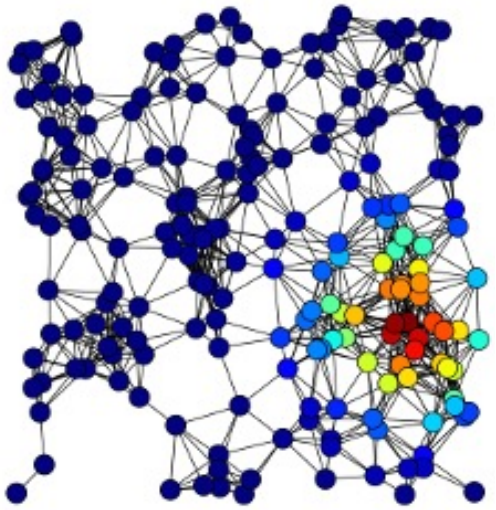
Betweeness centrality

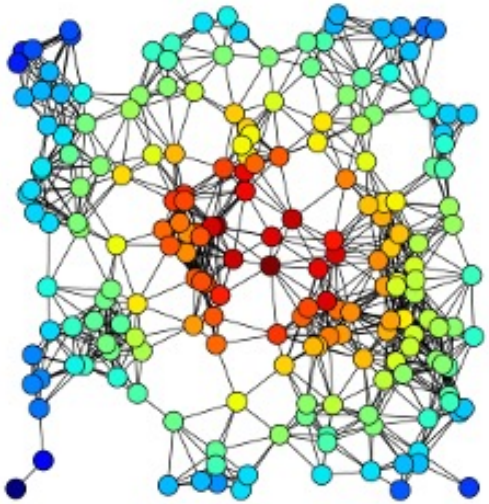
Same network, different metrics, different concepts.



Degree centrality

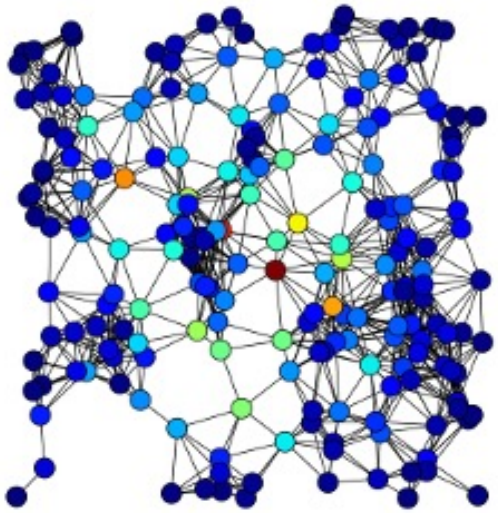
Eigenvector, centrality



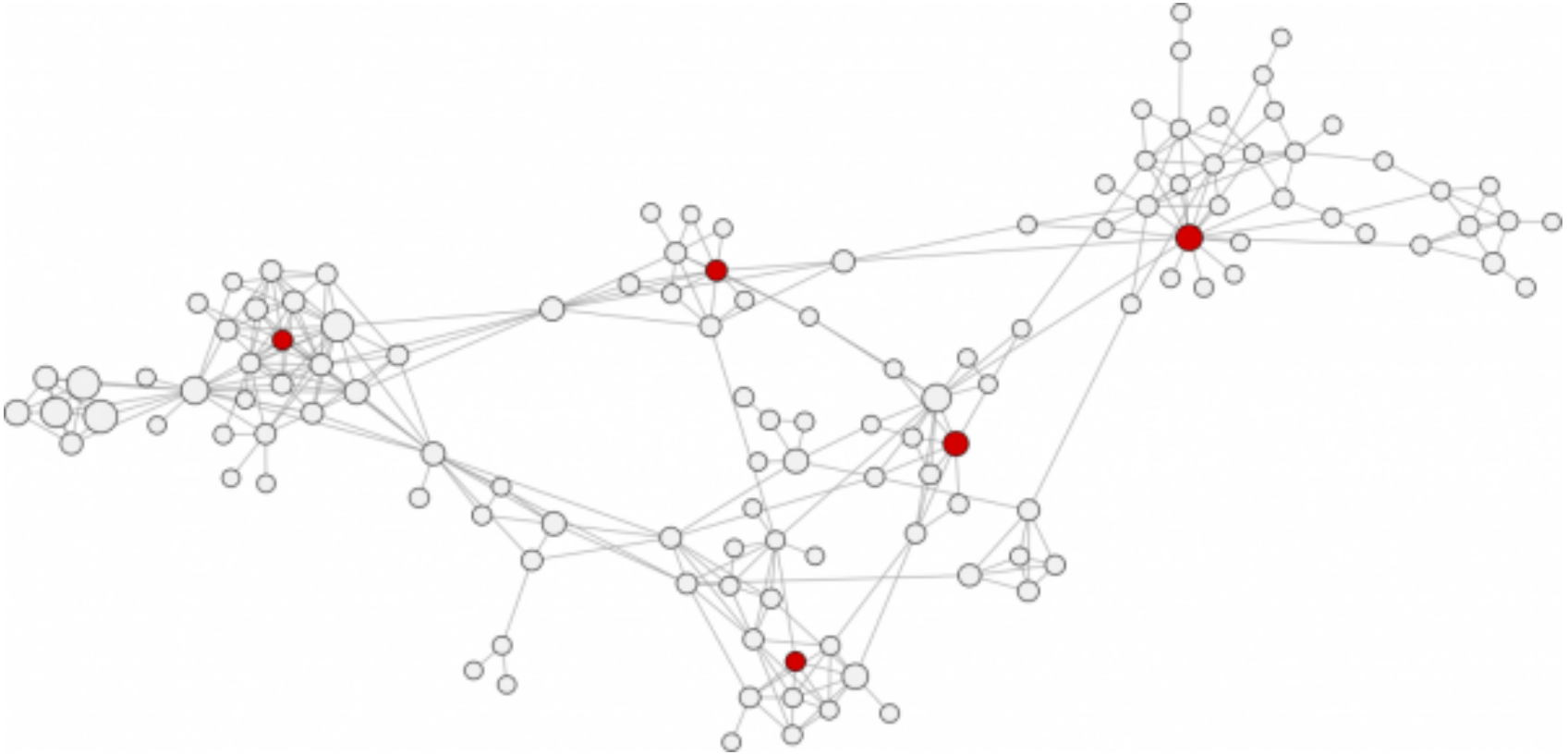


Closeness Centrality

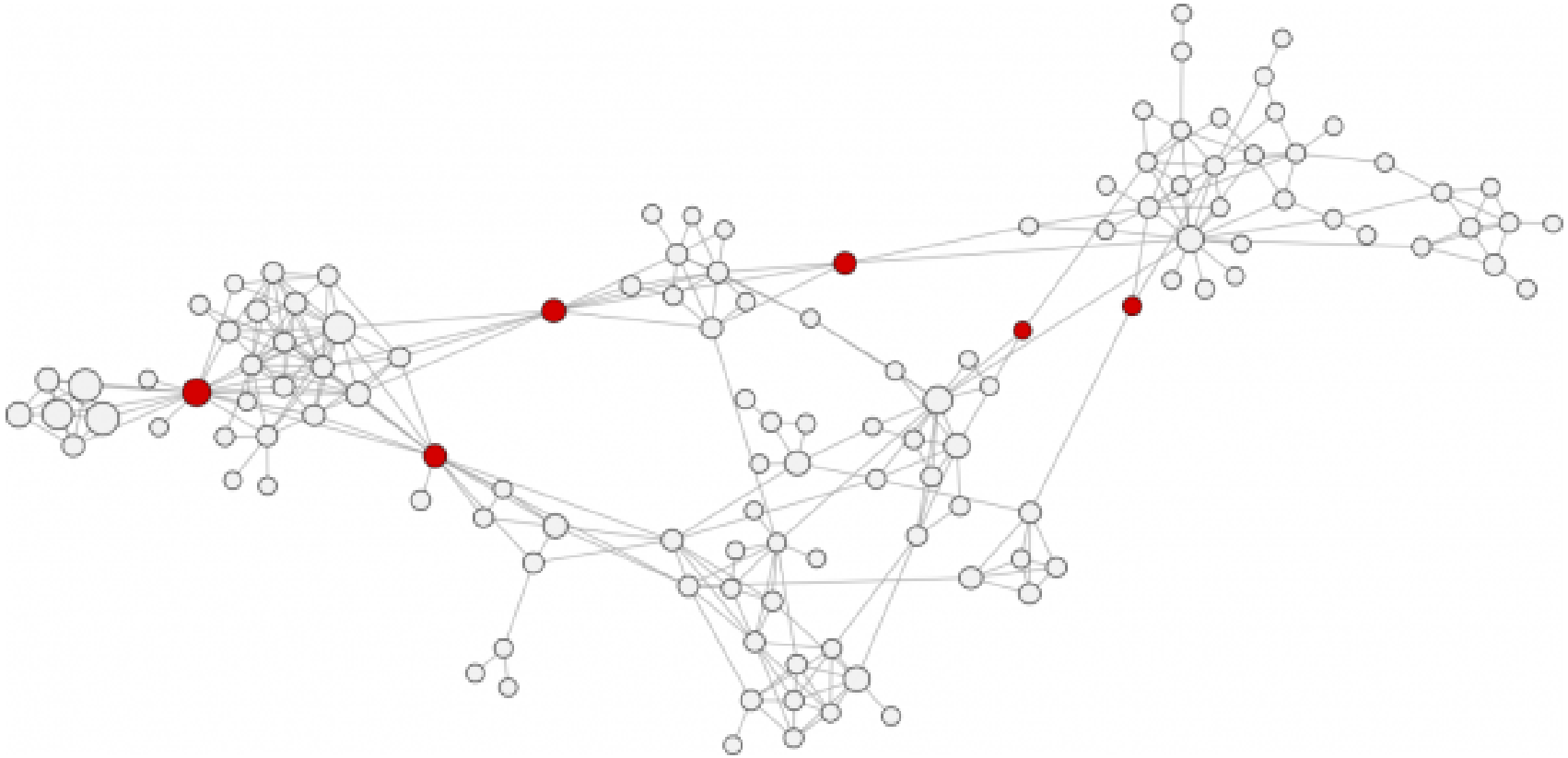
Betweeness centrality



Closeness centrality



Betweenness centrality



Eigenvector centrality

Gephi

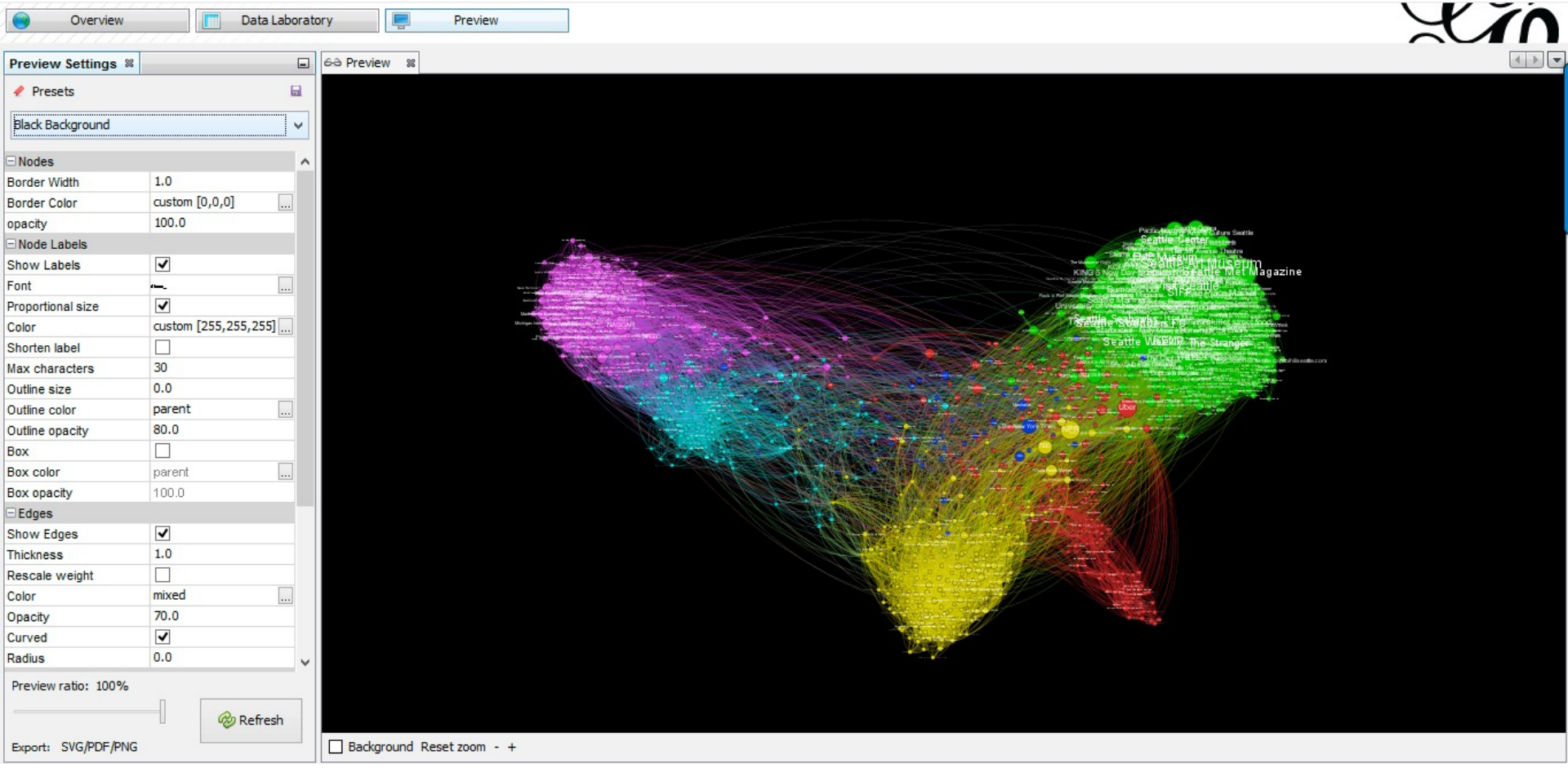
Boost

NetworkX
Network Analysis in Python

igraph

WebGraph
http://webgraph.di.unimi.it/

Mathematica

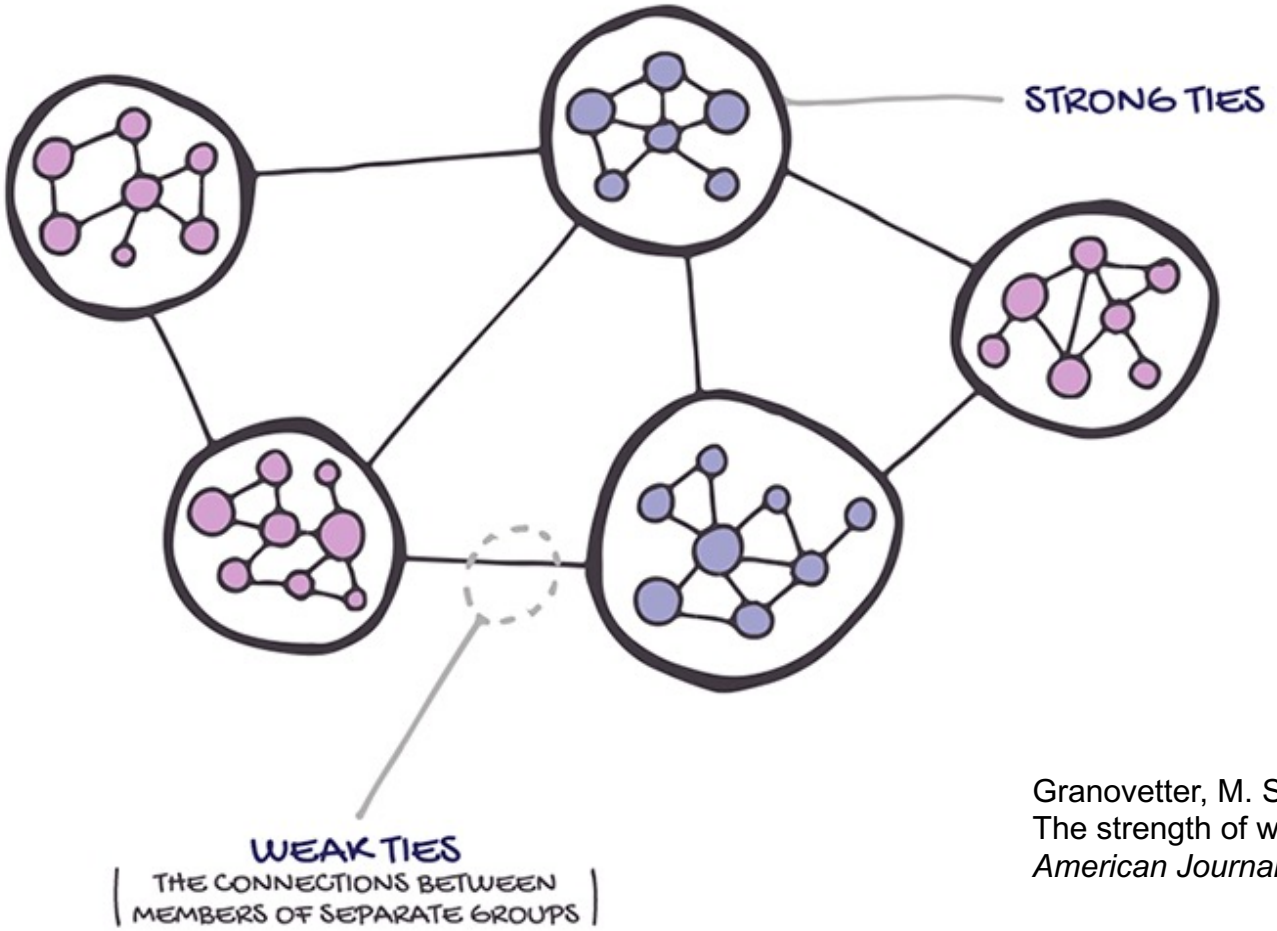


Often our focus should be on the edges…

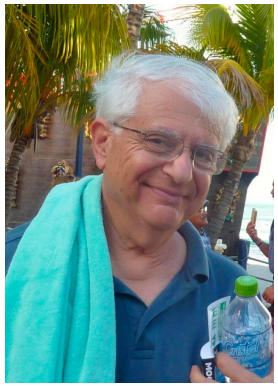
Which links are the most important or central in a network?

The strength of weak ties

90% got information about their current job not from the closest peers but from people that they do not know very well… their weak ties.



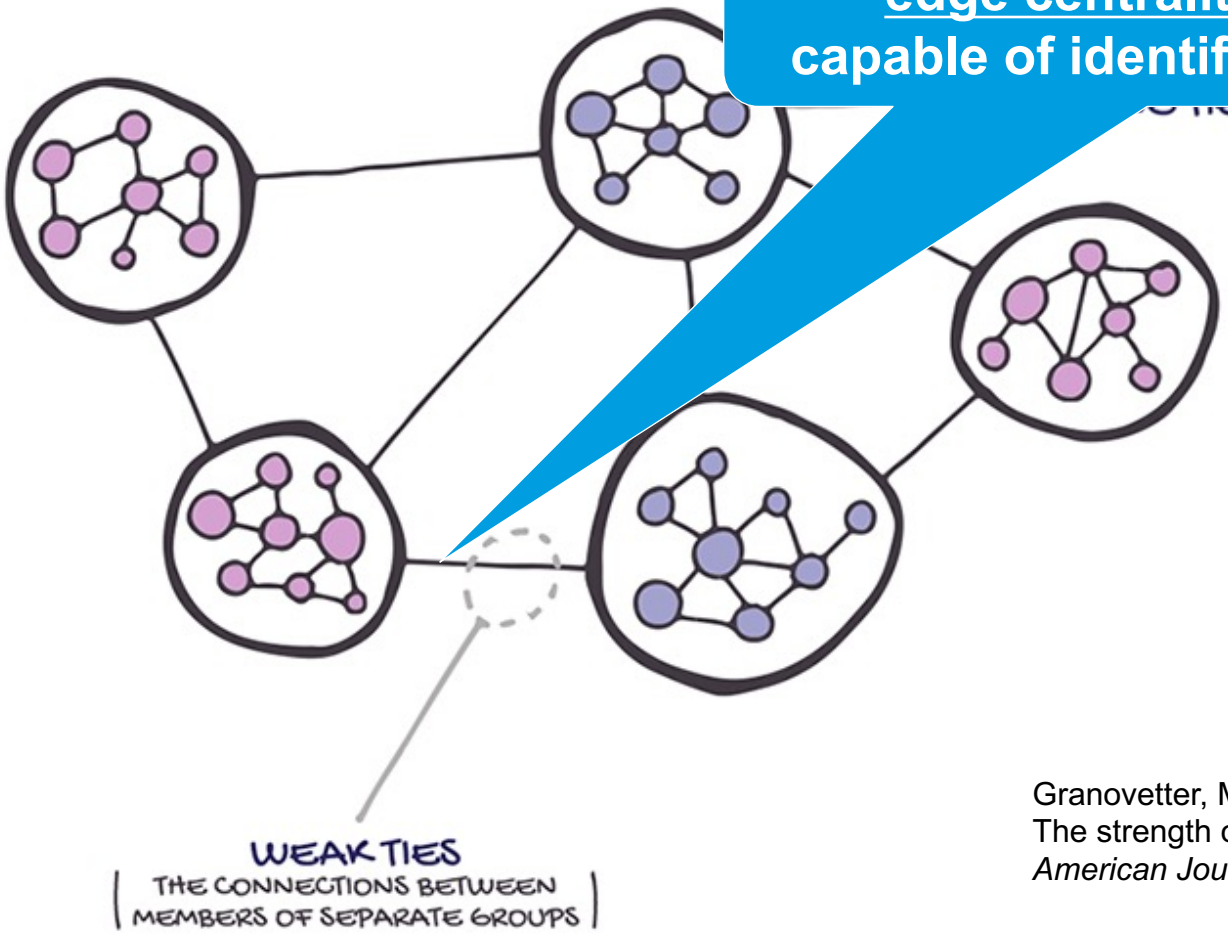
Granovetter, M. S. (1973).
The strength of weak ties.
American Journal of Sociology



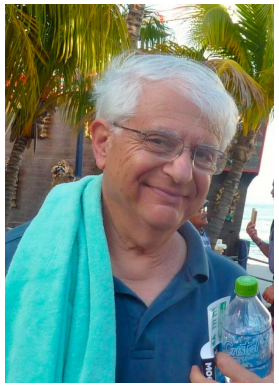
The strength of weak ties

Weak ties enable reaching populations and audiences that are not accessible via strong ties.

Propose an edge centrality measure capable of identifying weak ties

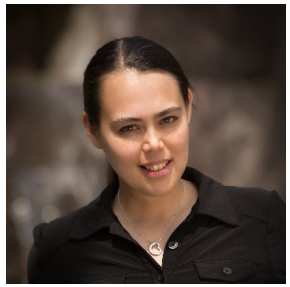


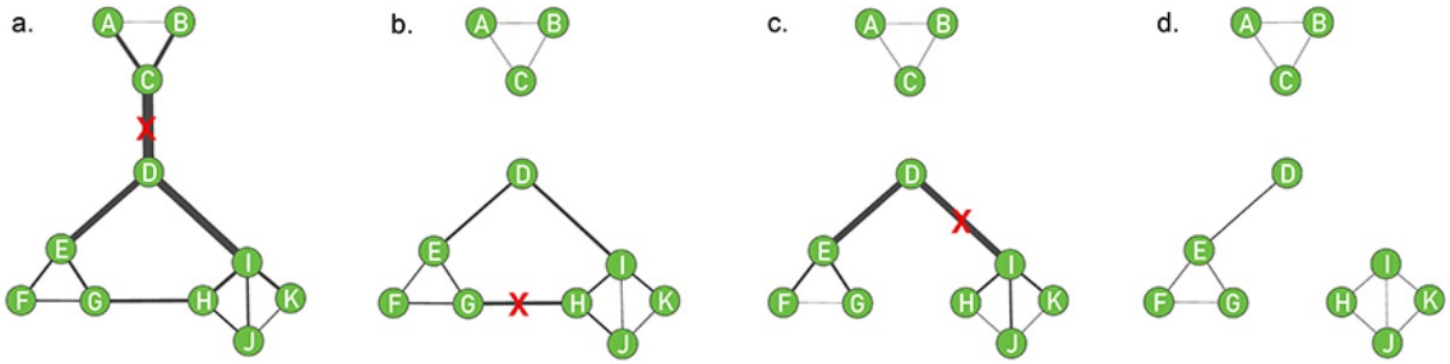
Granovetter, M. S. (1973).
The strength of weak ties.
American Journal of Sociology

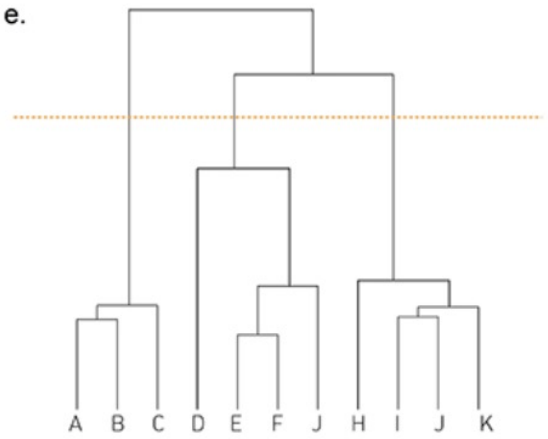


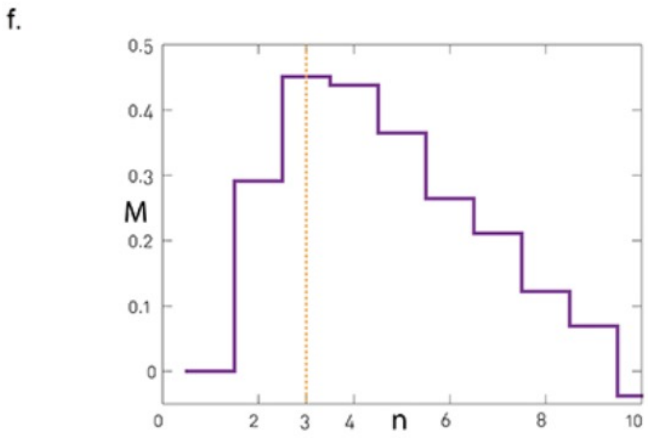
Weak ties & community finding

Girvan-Newman Community finding algorithm
(we will return to this later)





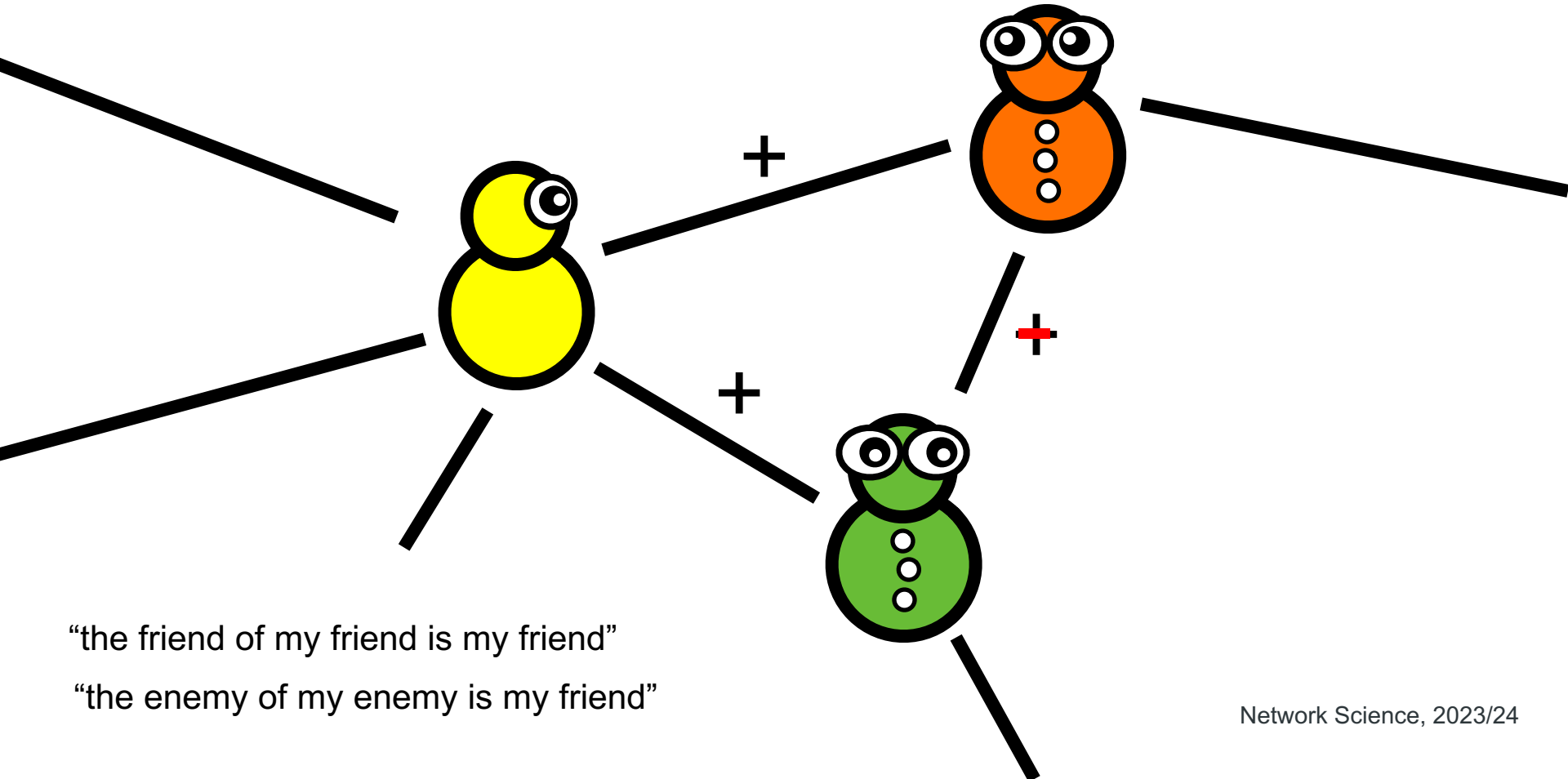




Other example…

Social balance theory

social networks often contain both friendly and unfriendly pairwise links between individual nodes



"the friend of my friend is my friend"

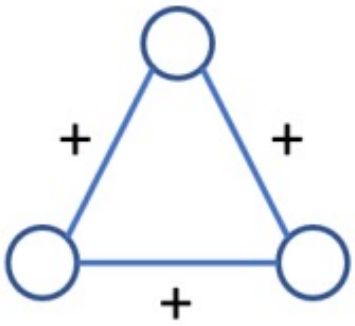
"the enemy of my enemy is my friend"

Other example…

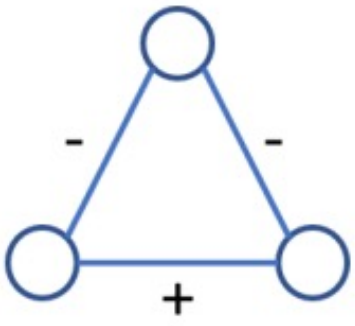
Social balance theory

social networks often contain both friendly and unfriendly pairwise links between individual nodes

unfrustrated

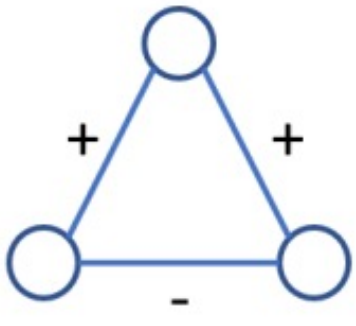


balanced

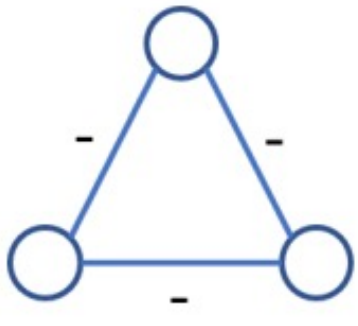


balanced

frustrated



unbalanced



unbalanced

"the friend of my friend is my friend"

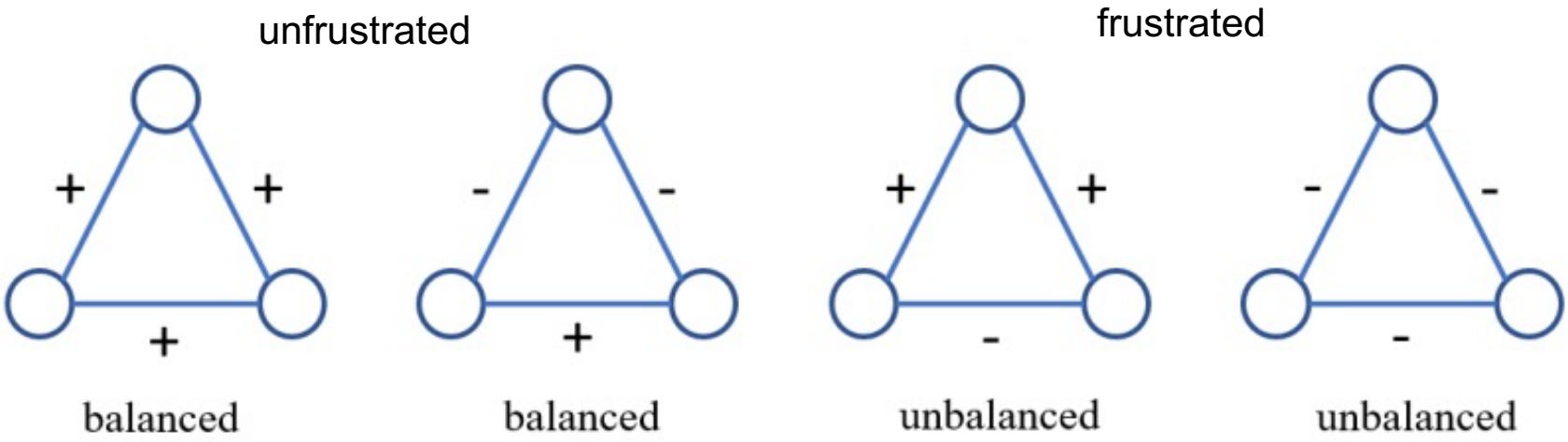
"the enemy of my enemy is my friend"

A triad is considered balanced if the product of the signs is positive

Other example…

Social balance theory

social networks often contain both friendly and unfriendly pairwise links between individual nodes



The social balance index of a network is the fraction of balanced triads in a network

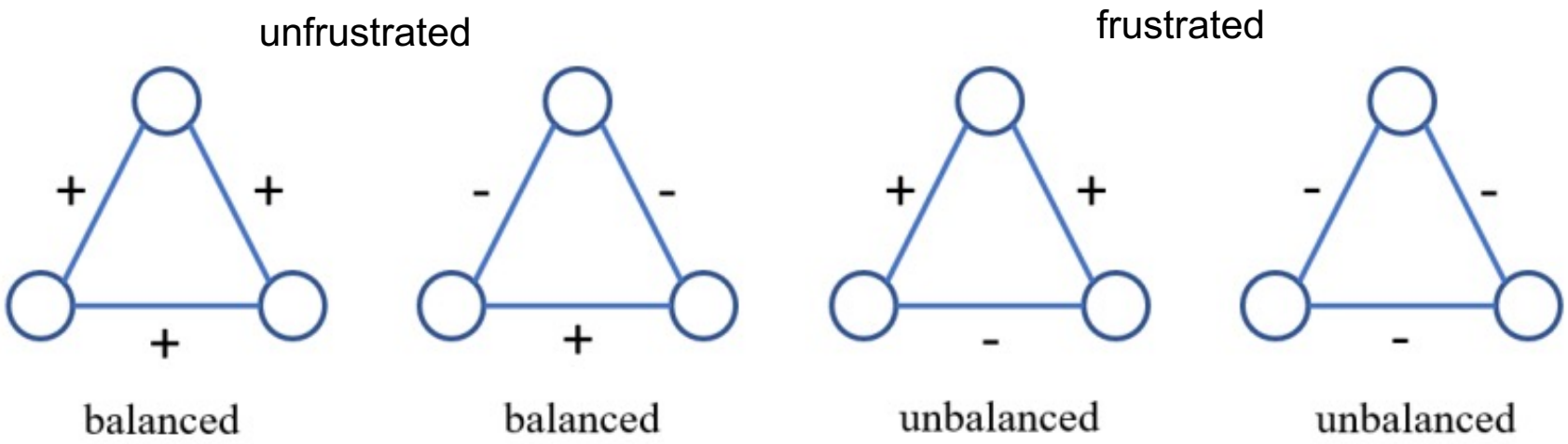
A network is balanced if each constituent triad is balanced

A seemingly more general definition of a balanced network is to require that each closed cycle is balanced

Other example…

Social balance theory

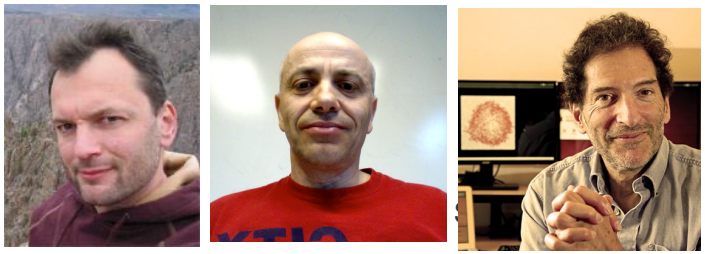
social networks often contain both friendly and unfriendly pairwise links between individual nodes



The social balance index of a network is the fraction of balanced triads in a network

Possible project: Which type of simple social/network dynamics give rise to a balanced network?

Antal, Krapivsky, Redner,
Dynamics of social balance on networks, Physical Review E, 2005
Social balance on networks: The dynamics of friendship and enmity, Physica D, 2006

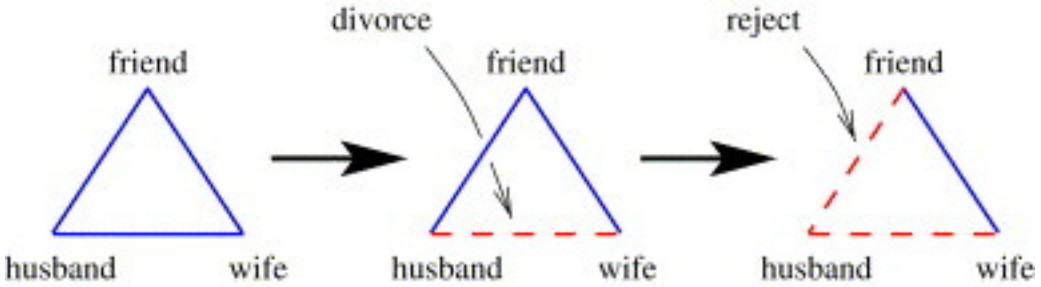


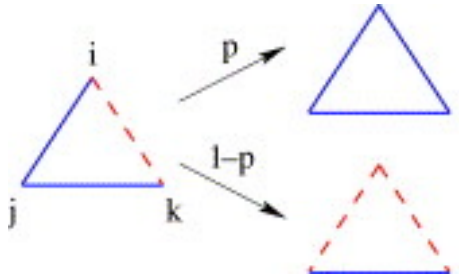
Tibor Antal, Pavel Krapivsky, and Sid Redner

Other example…

Social balance theory

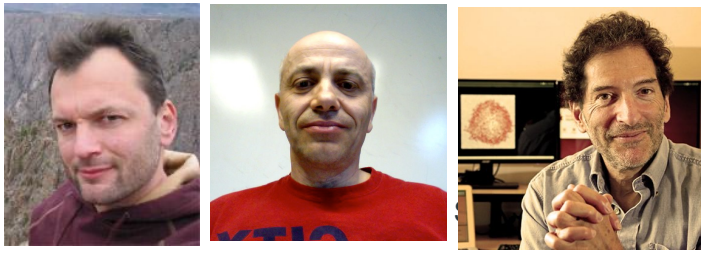
Examples:





Possible project: Which type of simple social/network dynamics give rise to a balanced network?

Antal, Krapivsky, Redner,
Dynamics of social balance on networks, Physical Review E, 2005
Social balance on networks: The dynamics of friendship and enmity, Physica D, 2006

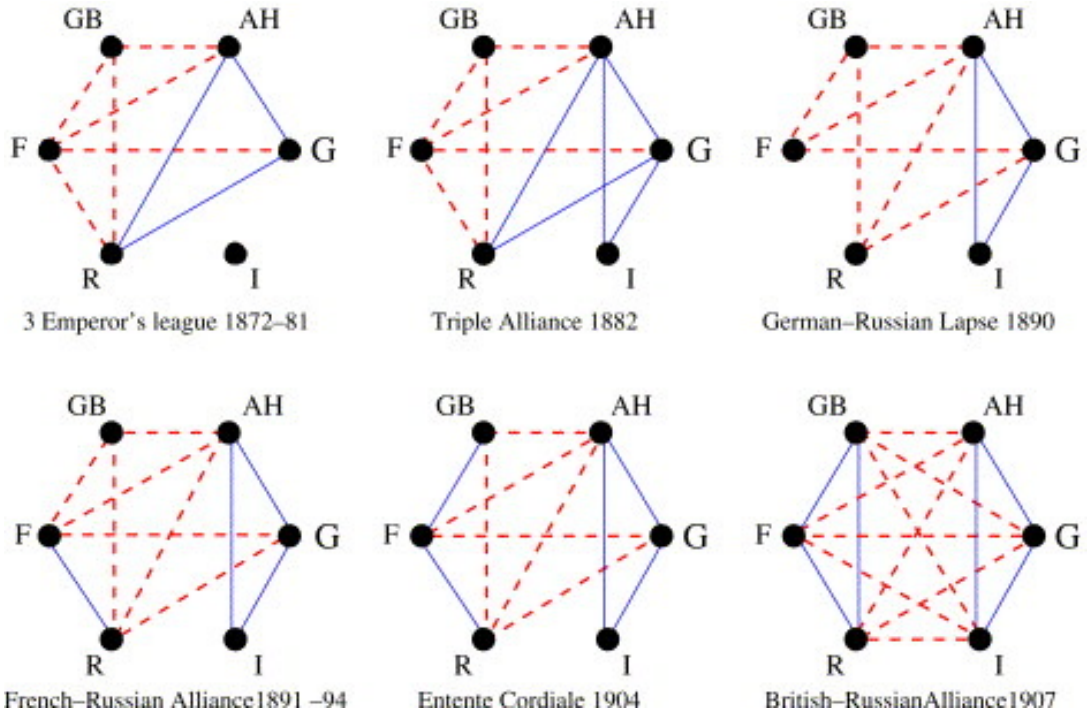


Tibor Antal, Pavel Krapivsky, and Sid Redner

Other example…

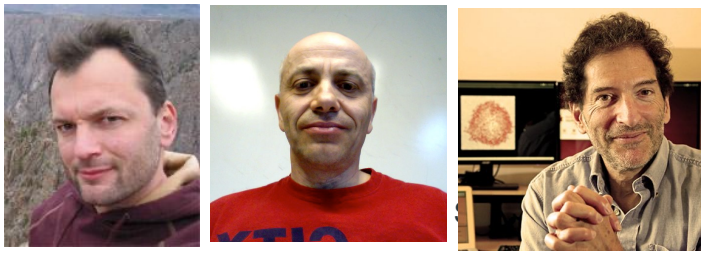
Social balance theory

Evolution of the major relationship changes between the protagonists of World War I from 1872–1907



Possible project: Which type of simple social/network dynamics give rise to a balanced network?

Antal, Krapivsky, Redner,
Dynamics of social balance on networks, Physical Review E, 2005
Social balance on networks: The dynamics of friendship and enmity, Physica D, 2006



Tibor Antal, Pavel Krapivsky, and Sid Redner

Other example…

Social balance theory & suicidal notes

Andreia Sofia Teixeira

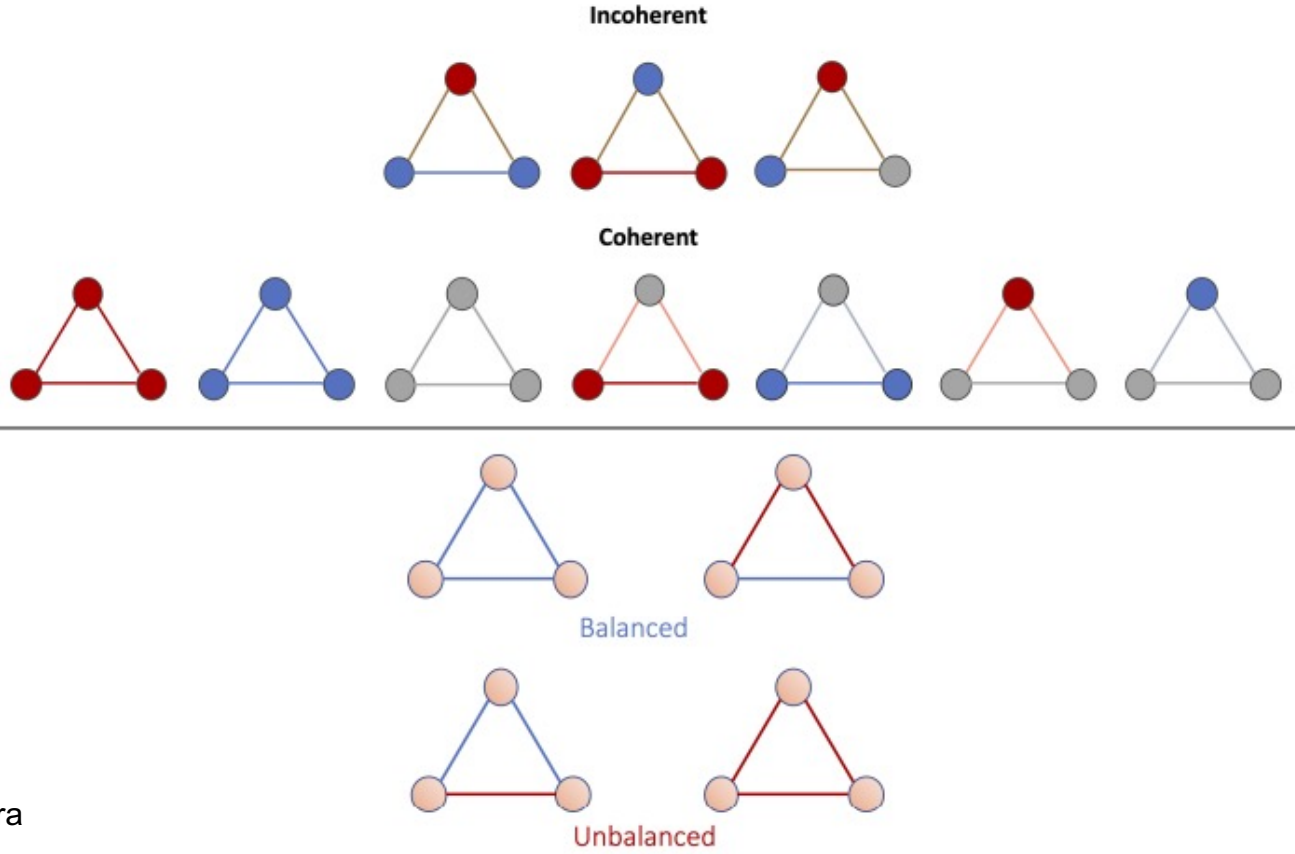


Figure 1: Comparison of emotional complexity in word networks (top) versus structural balance in social networks (bottom). Colours indicate positive (blue), negative (red) or neutral (gray) words (top) or social ties (bottom).

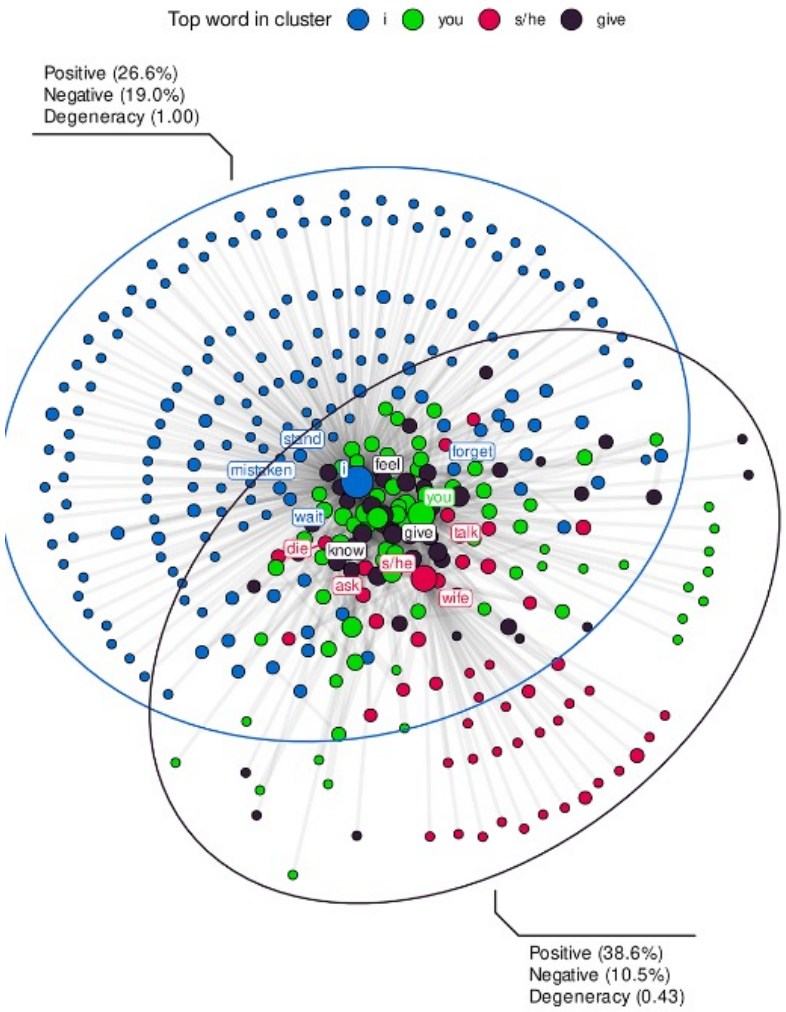
Teixeira, A. S., Talaga, S., Swanson, T. J., & Stella, M. (2021). Revealing semantic and emotional structure of suicide notes with cognitive network science. *Scientific Reports*.

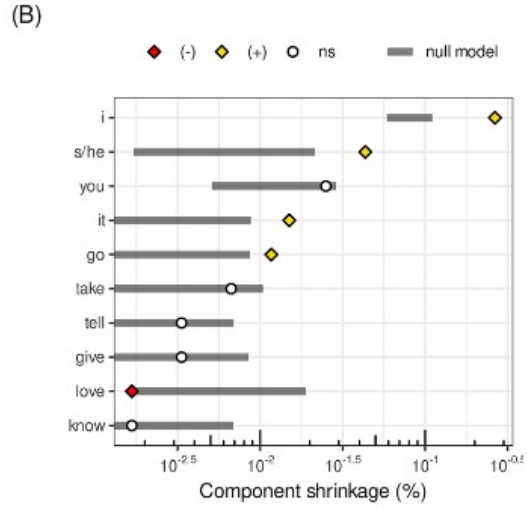
Stella, M., Swanson, T., Hills, T. T., & Teixeira, A. S. (2021). Cognitive network science as a framework for detecting structural patterns and emotions in suicide letters.

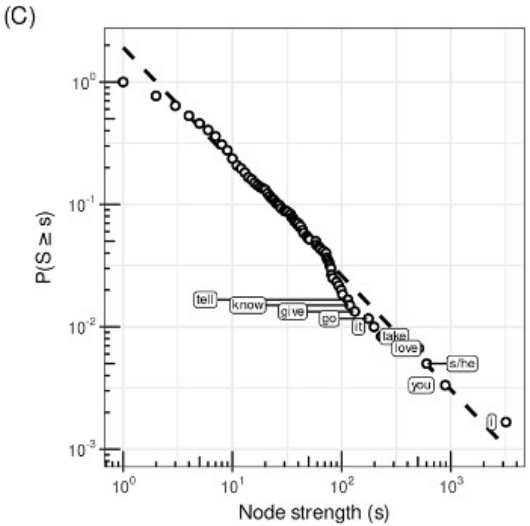
Other example…

Social balance theory & suicidal notes

Andreia Sofia Teixeira



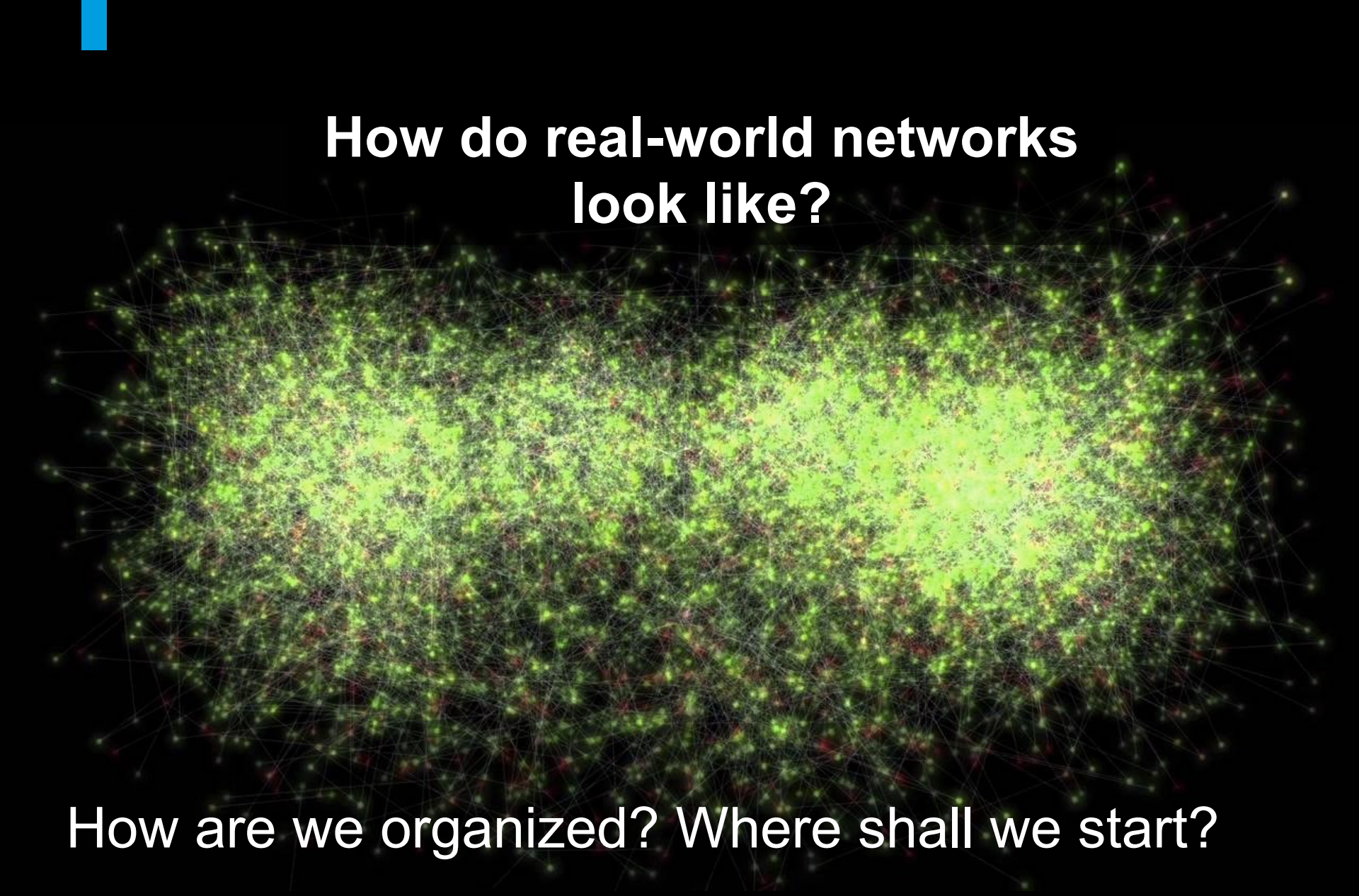




Teixeira, A. S., Talaga, S., Swanson, T. J., & Stella, M. (2021). Revealing semantic and emotional structure of suicide notes with cognitive network science. *Scientific Reports*.

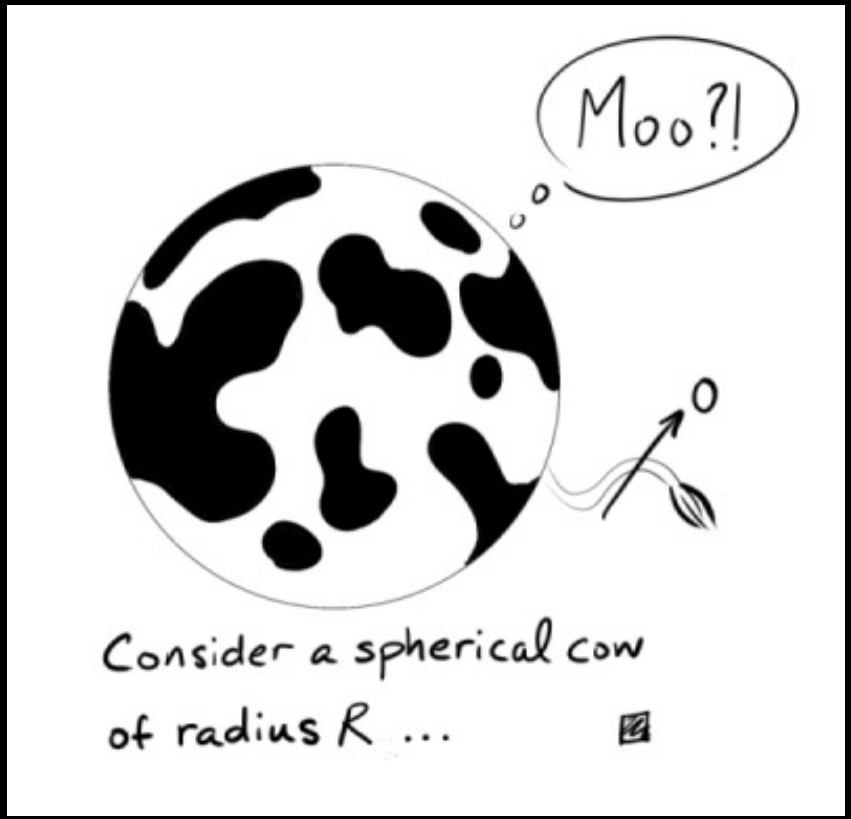
Stella, M., Swanson, T., Hills, T. T., & Teixeira, A. S. (2021). Cognitive network science as a framework for detecting structural patterns and emotions in suicide letters.

How do real-world networks look like?



How are we organized? Where shall we start?

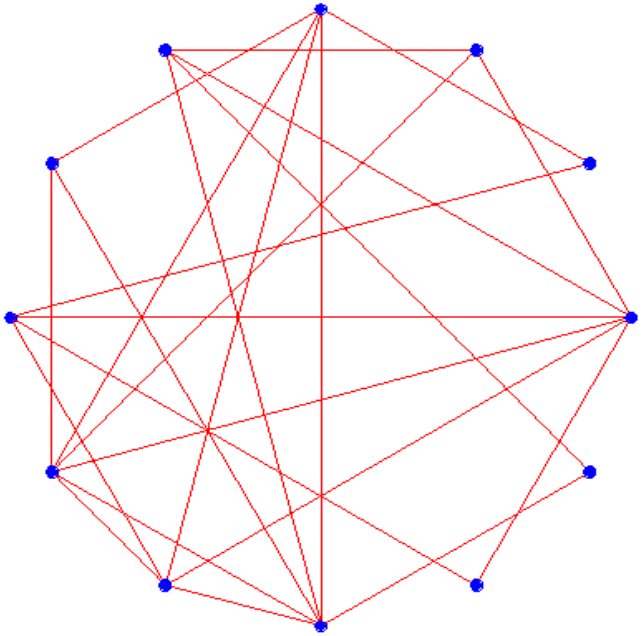
Modeling small-world properties? Where shall we start?



The simplest model one can think…

P. Erdős and A. Rényi, On Random Graphs, Publ. Math. 6, 290 (1959).

- A random network consists of *N* nodes where each pair of nodes is connected with probability ***p***.



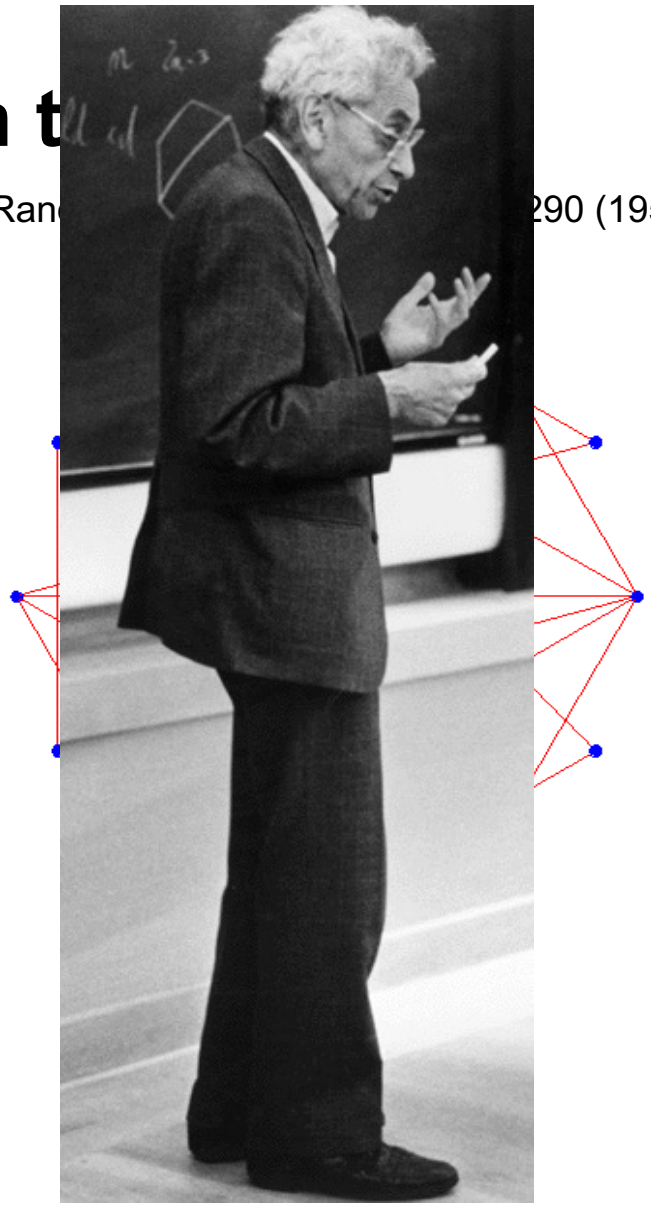
Random network model or the *Erdös and Renyi (ER) Model*

The null model of network science

The simplest model one can t

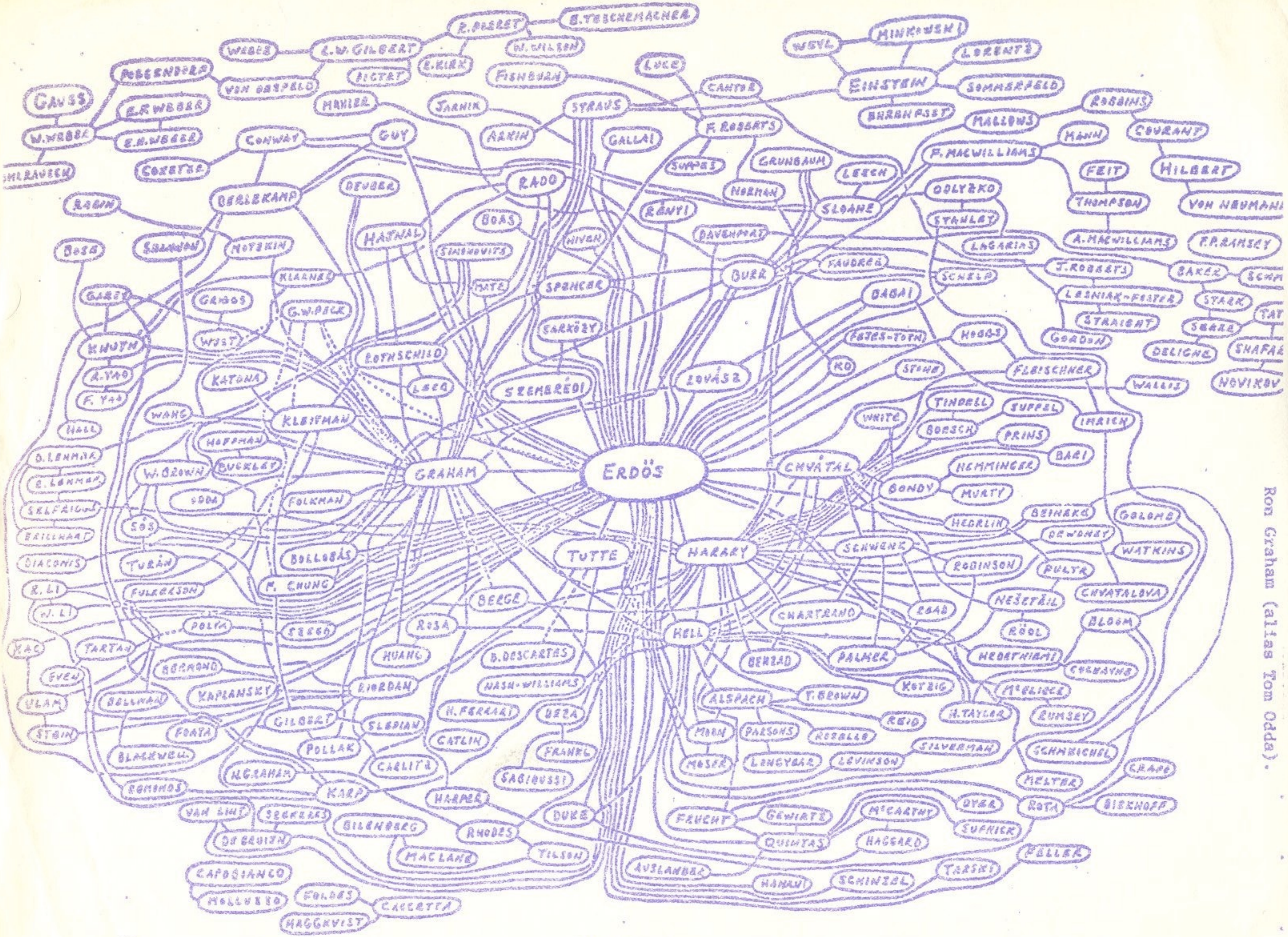
P. Erdős and A. Rényi, On Ran 290 (1959).

- Number of links?
- Average degree?
- Degree distribution?
- Clustering Coefficient?
- Small-world effect?
- Maximum degree?
- Giant component?



Random network model or the *Erdös and Renyi (ER) Model*

The null model of network science

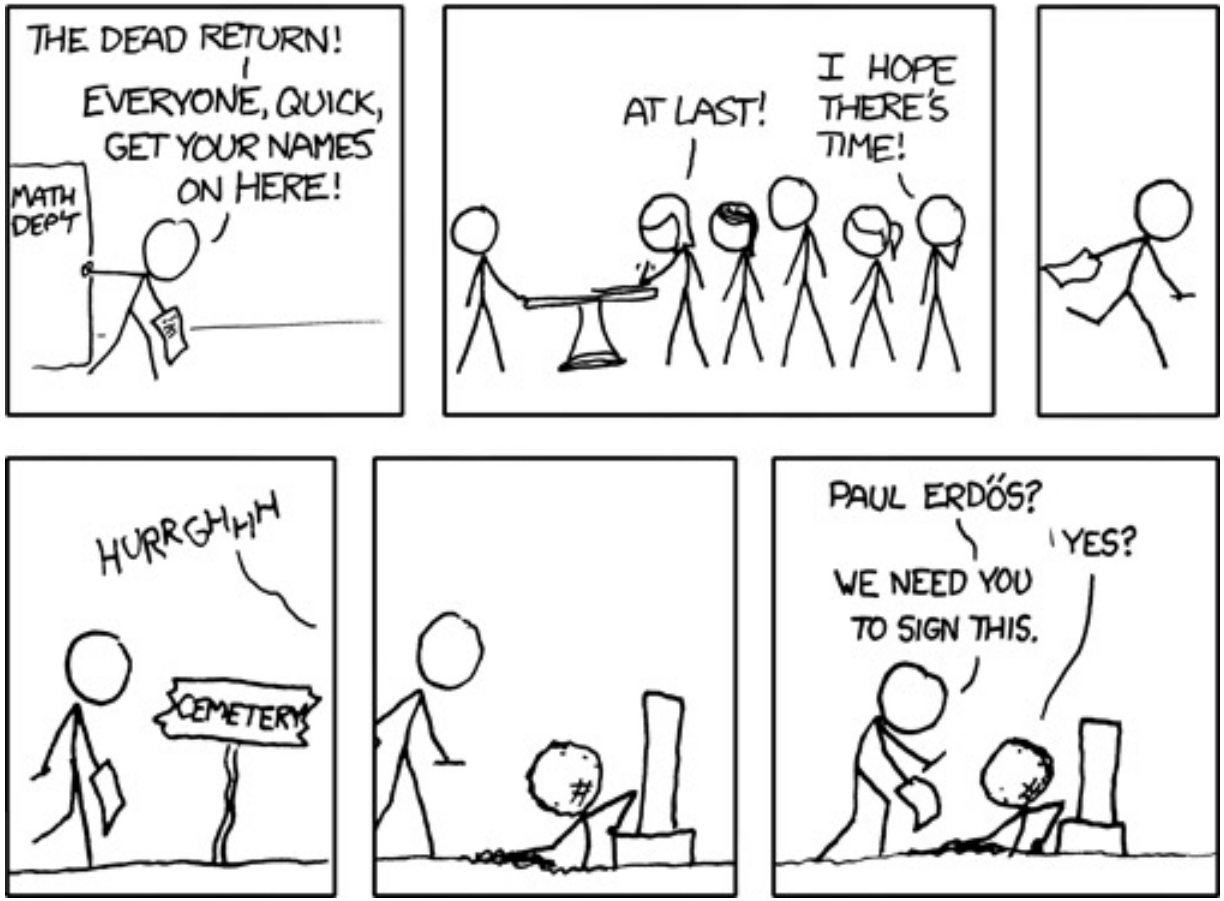


Ron Graham (alias Tom Odda).

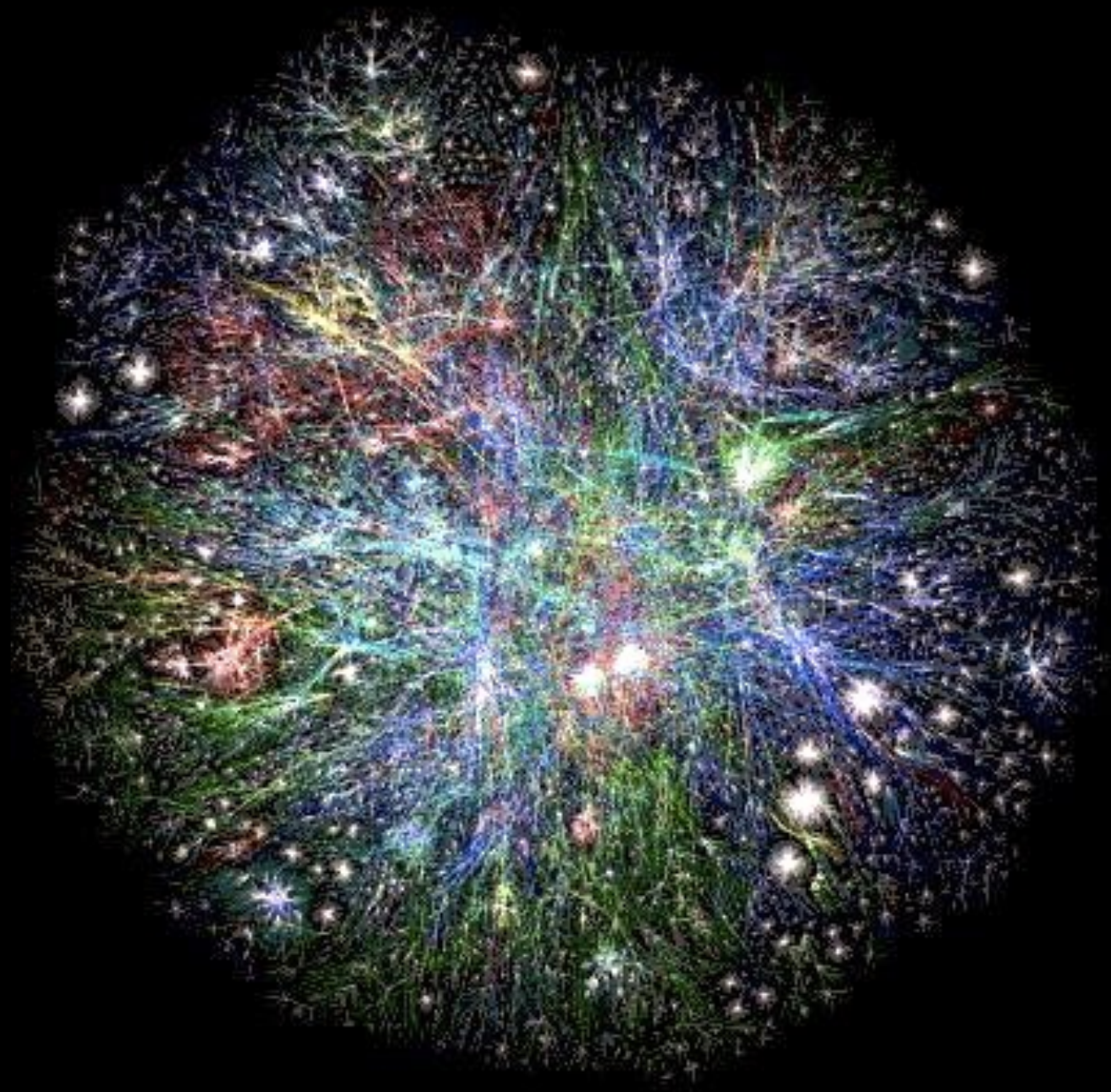
Figure 1

To appear in Topics in Graph Theory (F. Harary, ed.) New York Academy of Sciences (1979).

Science through Paul Erdös



Thank you!



GML format

Graph Modelling Language

(non-XML)

```
graph [
        comment "This is a sample graph"
        directed 1
        id n42
        label "Hello, I am a graph"
        node [
                id 1
                label "node 1"
                thisIsASampleAttribute 42
        ]
        node [
                id 2
                label "node 2"
                thisIsASampleAttribute 43
        ]
        node [
                id 3
                label "node 3"
                thisIsASampleAttribute 44
        ]
        edge [
                source 1
                target 2
                label "Edge from node 1 to node 2"
        ]
        edge [
                source 2
                target 3
                label "Edge from node 2 to node 3"
        ]
        edge [
                source 3
                target 1
                label "Edge from node 3 to node 1"
        ]
]
```

GEXF
(Graph Exchange XML Format)

http://gexf.net/data/hello-world.gexf

```
<?xml version="1.0" encoding="UTF-8"?>
<gexf xmlns="http://www.gexf.net/1.2draft" version="1.2">
    <meta lastmodifieddate="2009-03-20">
        <creator>Gexf.net</creator>
        <description>A hello world! file</description>
    </meta>
    <graph mode="static" defaultedgetype="directed">
        <nodes>
            <node id="0" label="Hello" />
            <node id="1" label="Word" />
        </nodes>
        <edges>
            <edge id="0" source="0" target="1" />
```

GraphML