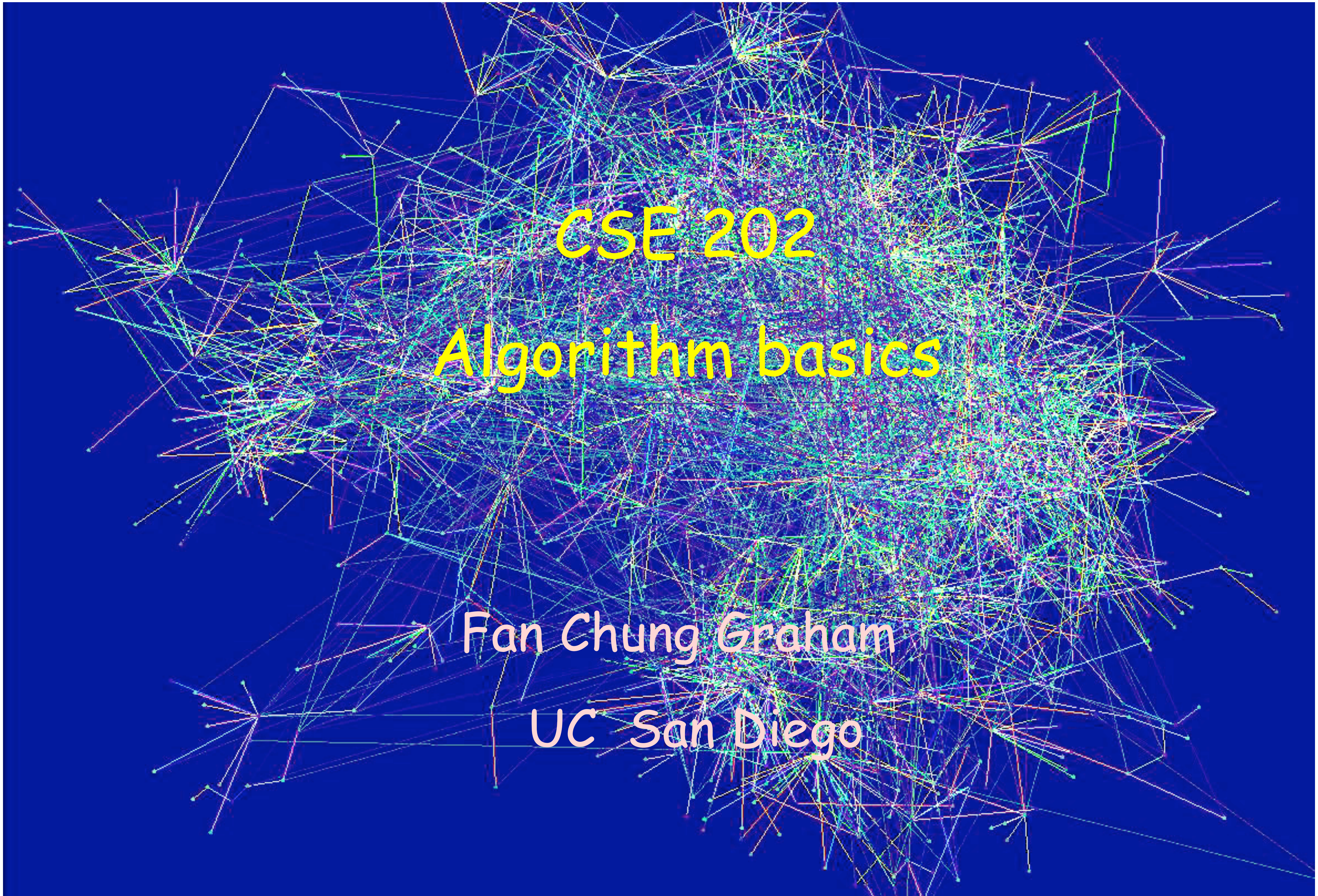
CSE 202

Algorithm basics

Fan Chung Graham

UC San Diego



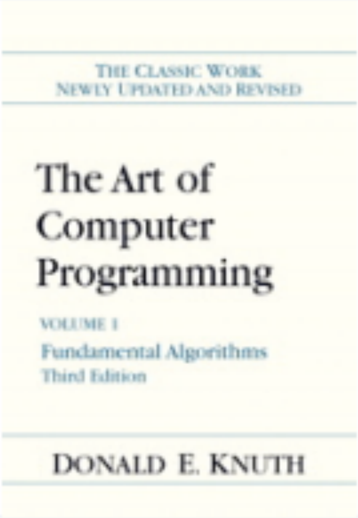
An induced subgraph of the collaboration graph (with Erdos number at most 2).
Made by Fan Chung Graham and Lincoln Lu in 2002.

What is an algorithm?

“ A procedure for solving a mathematical problem (as of finding the greatest common divisor) in a finite number of steps that frequently involves repetition of an operation. ” — *webster.com*

“ An algorithm is a finite, definite, effective procedure, with some input and some output. ”

— *Donald Knuth*



Algorithm

EXPLORE THIS TOPIC IN The *MathWorld* Classroom
CONTRIBUTE This Entry

An algorithm is a specific set of instructions for carrying out a procedure or solving a problem, usually with the requirement that the procedure terminate at some point. Specific algorithms sometimes also go by the name method, procedure, or technique. The word "algorithm" is a distortion of al-Khwārizmī, a Persian mathematician who wrote an influential treatise about algebraic methods. The process of applying an algorithm to an input to obtain an output is called a computation.

SEE ALSO: 196-Algorithm, Archimedes Algorithm, Brelaz's Heuristic Algorithm, Buchberger's Algorithm, Bulirsch-Stoer Algorithm, Bumping

Algorithm etymology

Etymology. [Knuth, TAOCP]

- *Algorism* = process of doing arithmetic using Arabic numerals.
- A misperception: *algiros* [painful] + *arithmos* [number].
- True origin: Abu 'Abd Allah Muhammad ibn Musa al-Khwarizm was a famous 9th century Persian textbook author who wrote *Kitāb al-jabr wa'l-muqābala*, which evolved into today's high school algebra text.


Abū ʿAbdallāh Muḥammad ibn Mūsā al-Khwārizmī (c. 780 – c. 850)

was a Persian mathematician, astronomer and geographer, a scholar in the House of Wisdom in Baghdad.

His *Kitab al-Jabr wa-l-Muqabala* presented the first systematic solution of linear and quadratic equations. He is considered the founder of algebra, a credit he shares with Diophantus. In the twelfth century, Latin translations of his work on the Indian numerals, introduced the decimal positional number system to the Western world. He revised Ptolemy's *Geography* and wrote on astronomy and astrology.

His contributions had a great impact on language. "**Algebra**" is derived from *al-jabr*, one of the two operations he used to solve quadratic equations. *Algorism* and *algorithm* stem from **Algoritmi**, the Latin form of his name.

From WIKI

Why study algorithms?

Internet. Web search, packet routing, distributed file sharing, ...

Biology. Human genome project, protein folding, ...

Computers. Circuit layout, databases, caching, networking, compilers, ...

Computer graphics. Movies, video games, virtual reality, ...

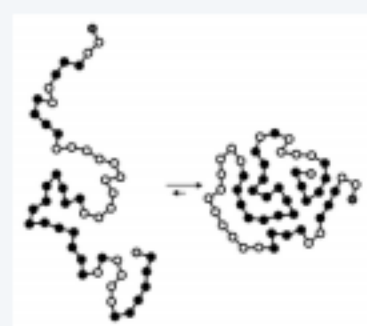
Security. Cell phones, e-commerce, voting machines, ...

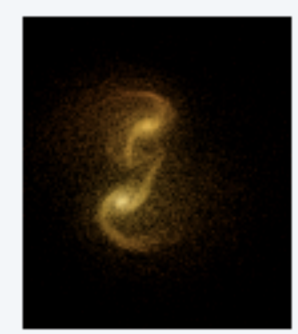
Multimedia. MP3, JPG, DivX, HDTV, face recognition, ...

Social networks. Recommendations, news feeds, advertisements, ...

Physics. N-body simulation, particle collision simulation, ...

⋮

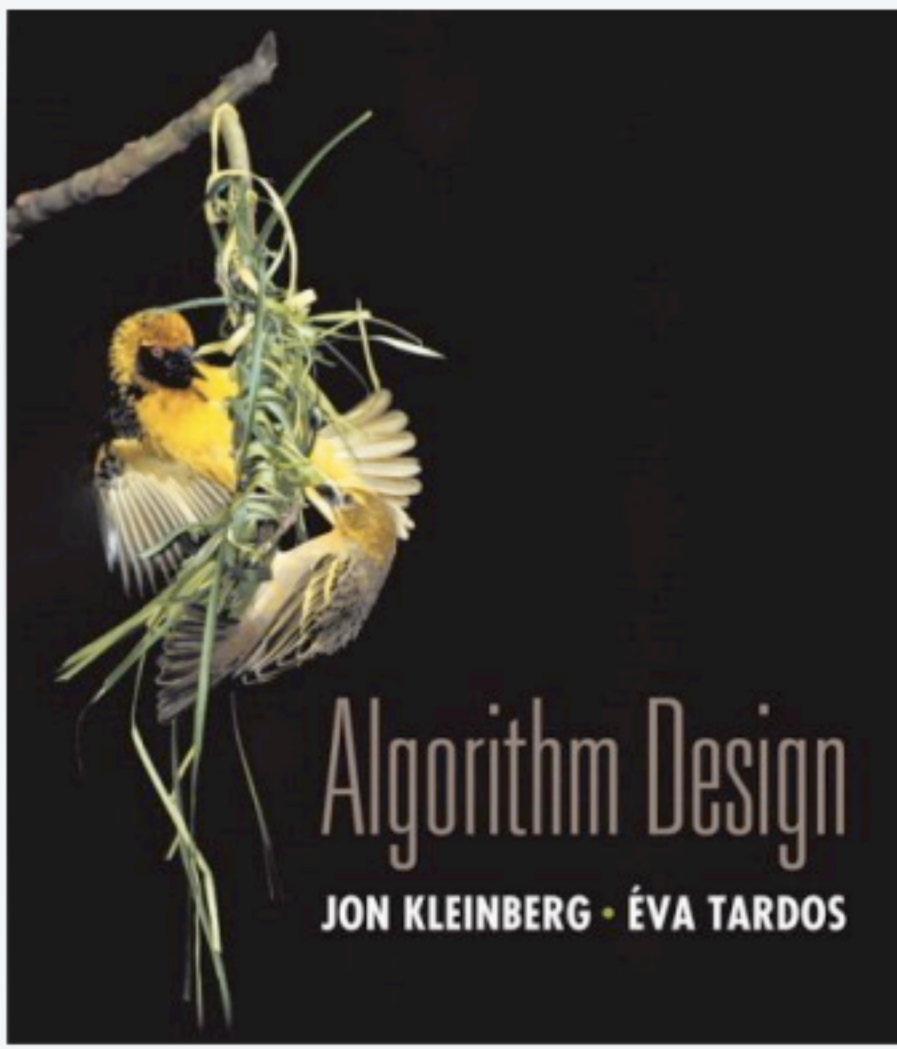





We emphasize algorithms and techniques that are useful in practice.

Textbook

Required reading. *Algorithm Design* by Jon Kleinberg and Éva Tardos.
Addison-Wesley 2005, ISBN 978-0321295354.




Euclidean algorithm:

Find the largest common factor between 36 and 123.

Euclidean algorithm:

Find the largest common factor between 36 and 123.

$$\begin{array}{r|l} & \,3 \\ \hline 36 & 123 \\ & 108 \\ \hline & 15 \end{array} \qquad 123 = 3(36)+15$$

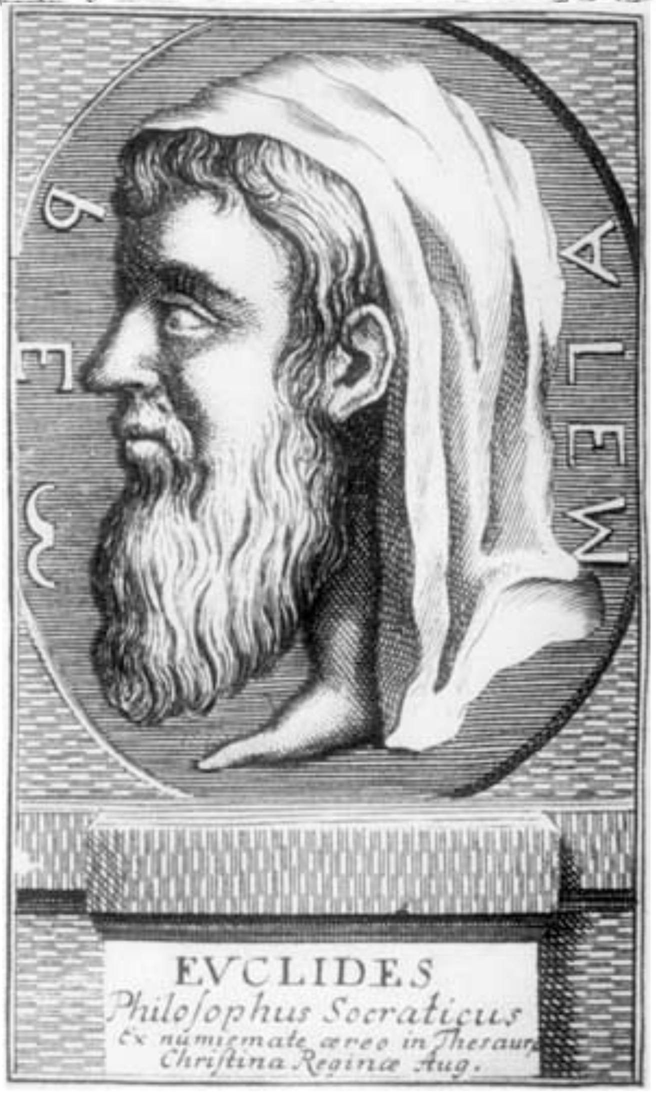
$$\begin{array}{r|l} & 2 \\ \hline 15 & 36 \\ & 30 \\ \hline & 6 \end{array} \qquad 36 = 2(15)+6$$

$$\begin{array}{r|l} & 2 \\ \hline 6 & 15 \\ & 12 \\ \hline & 3 \end{array} \qquad 15 = 2(6)+3$$

$$\begin{array}{r|l} & 2 \\ \hline 3 & 6 \\ & 6 \\ \hline & 0 \end{array} \qquad 6 = 2(3)$$

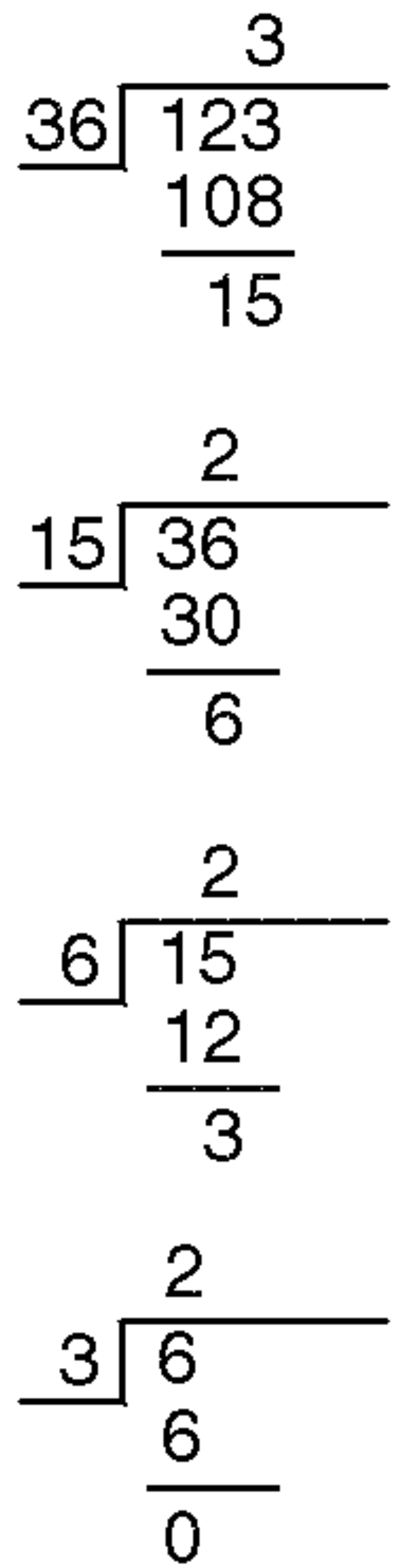
Euclidean of Alxandria (~325 BC)

Euclid of Alexandria is the most prominent mathematician of antiquity best known for his treatise on mathematics *The Elements*. The long lasting nature of *The Elements* must make Euclid the leading mathematics teacher of all time.



Euclidean algorithm:

Find the largest common factor between 36 and 123.



$$123 = 3(36)+15$$

$$36 = 2(15)+6$$

$$15 = 2(6)+3$$

$$6 = 2(3)$$

Euclidean algorithm:

Find the largest common factor between a and b.

Algorithm #1:

```
function gcd(a, b)
    while b ≠ 0
        t := b
        b := a mod b
        a := t
    return a
```

Algorithm #2:

```
function gcd(a, b)
    while a ≠ b
        if a > b
            a := a − b
        else
            b := b − a
    return a
```

Algorithm analysis:

☑ Termination?

☑ Correctness?

☑ Efficiency?

Emphasizes critical thinking, problem-solving

Algorithm Analysis

- Worst case running time.

- Average case running time.

Algorithm Analysis

- Worst case running time. Obtain bound on largest possible running time of algorithm on input of a given size N.
  - Generally captures efficiency in practice.
  - Draconian view, but hard to find effective alternative.

- Average case running time. Obtain bound on running time of algorithr on random input as a function of input size N.
  - Hard (or impossible) to accurately model real instances by random distributions.
  - Algorithm tuned for a certain distribution may perform poorly on other inputs.

Brute-Force Search

Brute force. For many non-trivial problems, there is a natural brute force search algorithm that checks every possible solution.

- Typically takes 2^N time or worse for inputs of size N.
- Unacceptable in practice.

n! for stable matching
with n men and n women

- Not only too slow to be useful, it is an intellectual cop-out.
- Provides us with absolutely no insight into the structure of the problem.

Proposed definition of efficiency. An algorithm is efficient if it achieves qualitatively better worst-case performance than brute-force search.

Polynomial-Time

Desirable scaling property. When the input size doubles, the algorithm should only slow down by some constant factor *C*.

> There exists constants $c > 0$ and $d > 0$ such that on every input of size N, its running time is bounded by cN^d steps.

A step. a single assembly-language instruction, one line of a programming language like *C*...

What happens if the input size increases from N to $2N$?

Def. An algorithm is poly-time if the above scaling property holds.

Worst-Case Polynomial-Time

Def. An algorithm is efficient if its running time is polynomial.

Justification: It really works in practice!

- Although $6.02 \times 10^{23} \times N^{20}$ is technically poly-time, it would be useless in practice.
- In practice, the poly-time algorithms that people develop almost always have low constants and low exponents.
- Breaking through the exponential barrier of brute force typically exposes some crucial structure of the problem.

Exceptions.

- Some poly-time algorithms do have high constants and/or exponents, and are useless in practice.
- Some exponential-time (or worse) algorithms are widely used

Asymptotic Order of Growth

- We try to express that an algorithm's worst case running time is at most proportional to some function f(n).
- Function f(n) becomes a bound on the running time of the algorithm.

- Pseudo-code style.
  - counting the number of pseudo-code steps.
  - step. Assigning a value to a variable, looking up an entry in an array, following a pointer, a basic arithmetic operation…

"On any input size n, the algorithm runs for at most $1.62n^2 + 3.5n + 8$ steps."

Do we need such precise bound?

Asymptotic Order of Growth

Upper bounds. $T(n)$ is $O(f(n))$ if there exist constants $c > 0$ and $n_0 \geq 0$ such that for all $n \geq n_0$ we have $T(n) \leq c \cdot f(n)$.

Lower bounds. $T(n)$ is $\Omega(f(n))$ if there exist constants $c > 0$ and $n_0 \geq 0$ such that for all $n \geq n_0$ we have $T(n) \geq c \cdot f(n)$.

Tight bounds. $T(n)$ is $\Theta(f(n))$ if $T(n)$ is both $O(f(n))$ and $\Omega(f(n))$.

Ex: $T(n) = 32n^2 + 17n + 32$.

- $T(n)$ is $O(n^2)$, $O(n^3)$, $\Omega(n^2)$, $\Omega(n)$, and $\Theta(n^2)$.
- $T(n)$ is not $O(n)$, $\Omega(n^3)$, $\Theta(n)$, or $\Theta(n^3)$.

Asymptotic order of growth

$f_1(n) = O(f_2(n))$ means $f_1(n) < c\ f_2(n)$

as $n \to \infty$

$g_1(n) = \Omega(g_2(n))$ means $g_1(n) > c\ g_2(n)$

as $n \to \infty$

$h_1(n) = O(h_2(n))$ means $h_1(n) < c\ h_2(n) < c' h_1(n)$

as $n \to \infty$

where c and c' are absolute constants.

Arrange the following list of functions in ascending order of growth rate:

$$f_1(n) = n^{2.5}$$

$$f_2(n) = \sqrt{2n}$$

$$f_3(n) = n + 10$$

$$f_4(n) = 10^n$$

$$f_5(n) = 100^n$$

$$f_6(n) = n^2 \log n$$

complexity \ size	10	20	30	40	50	100	1000
$100n$	1 μsec	2 μsec	3 μsec	4 μsec	5 μsec	10 μsec	100 μsec
n^2	.1 μsec	.4 μsec	.9 μsec	1.6 μsec	2.5 μsec	10 μsec	1 msec
n^3							
2^n							
$n!$							

Polynomial vs. **exponential** growth

(assuming 1,000,000,000 operations per second)

n

size / complexity	10	20	30	40	50	100	1000
$100n$	1 μsec	2 μsec	3 μsec	4 μsec	5 μsec	10 μsec	100 μsec
n^2	.1 μsec	.4 μsec	.9 μsec	1.6 μsec	2.5 μsec	10 μsec	1 msec
n^3	1 μsec	8 μsec	27 μsec	64 μsec	.13 msec		
2^n							
$n!$							

Polynomial vs. **exponential** growth

(assuming 1,000,000,000 operations per second)

complexity \ size	10	20	30	40	50	100	1000
100n	1 μsec	2 μsec	3 μsec	4 μsec	5 μsec	10 μsec	100 μsec
n^2	.1 μsec	.4 μsec	.9 μsec	1.6 μsec	2.5 μsec	10 μsec	1 msec
n^3	1 μsec	8 μsec	27 μsec	64 μsec	.13 msec	1 msec	1 sec
2^n							
n!							

Polynomial vs. **exponential** growth

(assuming 1,000,000,000 operations per second)

complexity \ size	10	20	30	40	50	100	1000
100n	1 μsec	2 μsec	3 μsec	4 μsec	5 μsec	10 μsec	100 μsec
n^2	.1 μsec	.4 μsec	.9 μsec	1.6 μsec	2.5 μsec	10 μsec	1 msec
n^3	1 μsec	8 μsec	27 μsec	64 μsec	.13 msec	1 msec	1 sec
2^n	1 μsec	1 msec	1.1 sec				
n!							

Polynomial vs. **exponential** growth

(assuming 1,000,000,000 operations per second)

complexity \ size	10	20	30	40	50	100	1000
100n	1 μsec	2 μsec	3 μsec	4 μsec	5 μsec	10 μsec	100 μsec
n^2	.1 μsec	.4 μsec	.9 μsec	1.6 μsec	2.5 μsec	10 μsec	1 msec
n^3	1 μsec	8 μsec	27 μsec	64 μsec	.13 msec	1 msec	1 sec
2^n	1 μsec	1 msec	1.1 sec	18.3 min			
n!							

Polynomial vs. **exponential** growth

(assuming 1,000,000,000 operations per second)

complexity \ size (n)	10	20	30	40	50	100	1000
$100n$	1 μsec	2 μsec	3 μsec	4 μsec	5 μsec	10 μsec	100 μsec
n^2	.1 μsec	.4 μsec	.9 μsec	1.6 μsec	2.5 μsec	10 μsec	1 msec
n^3	1 μsec	8 μsec	27 μsec	64 μsec	.13 msec	1 msec	1 sec
2^n	1 μsec	1 msec	1.1 sec	18.3 min	2.1 yr		
$n!$							

Polynomial vs. **exponential** growth

(assuming 1,000,000,000 operations per second)

complexity \ size (n)	10	20	30	40	50	100	1000
100n	1 μsec	2 μsec	3 μsec	4 μsec	5 μsec	10 μsec	100 μsec
n^2	.1 μsec	.4 μsec	.9 μsec	1.6 μsec	2.5 μsec	10 μsec	1 msec
n^3	1 μsec	8 μsec	27 μsec	64 μsec	.13 msec	1 msec	1 sec
2^n	1 μsec	1 msec	1.1 sec	18.3 min	2.1 yr	2.4×10^{13} cent	
$n!$							

Polynomial vs. **exponential** growth

(assuming 1,000,000,000 operations per second)

complexity \ size	10	20	30	40	50	100	1000
100n	1 μsec	2 μsec	3 μsec	4 μsec	5 μsec	10 μsec	100 μsec
n^2	.1 μsec	.4 μsec	.9 μsec	1.6 μsec	2.5 μsec	10 μsec	1 msec
n^3	1 μsec	8 μsec	27 μsec	64 μsec	.13 msec	1 msec	1 sec
2^n	1 μsec	1 msec	1.1 sec	18.3 min	2.1 yr	2.4×10^{13} cent	----
$n!$							

Polynomial vs. **exponential** growth

(assuming 1,000,000,000 operations per second)

complexity \ size n	10	20	30	40	50	100	1000
$100n$	1 μsec	2 μsec	3 μsec	4 μsec	5 μsec	10 μsec	100 μsec
n^2	.1 μsec	.4 μsec	.9 μsec	1.6 μsec	2.5 μsec	10 μsec	1 msec
n^3	1 μsec	8 μsec	27 μsec	64 μsec	.13 msec	1 msec	1 sec
2^n	1 μsec	1 msec	1.1 sec	18.3 min	2.1 yr	2.4×10^{13} cent	----
$n!$	3.6 msec	1.8 yr					

Polynomial vs. **exponential** growth

(assuming 1,000,000,000 operations per second)

complexity \ size (n)	10	20	30	40	50	100	1000
100n	1 μsec	2 μsec	3 μsec	4 μsec	5 μsec	10 μsec	100 μsec
n^2	.1 μsec	.4 μsec	.9 μsec	1.6 μsec	2.5 μsec	10 μsec	1 msec
n^3	1 μsec	8 μsec	27 μsec	64 μsec	.13 msec	1 msec	1 sec
2^n	1 μsec	1 msec	1.1 sec	18.3 min	2.1 yr	2.4×10^{13} cent	----
n!	3.6 msec	1.8 yr	2.0×10^{15} cent				

Polynomial vs. **exponential** growth

(assuming 1,000,000,000 operations per second)

complexity \ size	10	20	30	40	50	100	1000
$100n$	1 μsec	2 μsec	3 μsec	4 μsec	5 μsec	10 μsec	100 μsec
n^2	.1 μsec	.4 μsec	.9 μsec	1.6 μsec	2.5 μsec	10 μsec	1 msec
n^3	1 μsec	8 μsec	27 μsec	64 μsec	.13 msec	1 msec	1 sec
2^n	1 μsec	1 msec	1.1 sec	18.3 min	2.1 yr	2.4×10^{13} cent	----
$n!$	3.6 msec	1.8 yr	2.0×10^{15} cent	!			

Polynomial vs. **exponential** growth

(assuming 1,000,000,000 operations per second)

complexity \ size	n = 10	20	30	40	50	100	1000
100n	1 μsec	2 μsec	3 μsec	4 μsec	5 μsec	10 μsec	100 μsec
n^2	.1 μsec	.4 μsec	.9 μsec	1.6 μsec	2.5 μsec	10 μsec	1 msec
n^3	1 μsec	8 μsec	27 μsec	64 μsec	.13 msec	1 msec	1 sec
2^n	1 μsec	1 msec	1.1 sec	18.3 min	2.1 yr	2.4×10^{13} cent	----
n!	3.6 msec	1.8 yr	2.0×10^{15} cent	!	forget it		

Polynomial vs. **exponential** growth

(assuming 1,000,000,000 operations per second)

size / complexity	n = 10	20	30	40	50	100	1000
100 n	1 μsec	2 μsec	3 μsec	4 μsec	5 μsec	10 μsec	100 μsec
n^2	.1 μsec	.4 μsec	.9 μsec	1.6 μsec	2.5 μsec	10 μsec	1 msec
n^3	1 μsec	8 μsec	27 μsec	64 μsec	.13 msec	1 msec	1 sec
2^n	1 μsec	1 msec	1.1 sec	18.3 min	2.1 yr	2.4×10^{13} cent	----
n!	3.6 msec	1.8 yr	2.0×10^{15} cent	!	forget it	?	

Polynomial vs. **exponential** growth

(assuming 1,000,000,000 operations per second)

complexity \ size (n)	10	20	30	40	50	100	1000
100 n	1 μsec	2 μsec	3 μsec	4 μsec	5 μsec	10 μsec	100 μsec
n^2	.1 μsec	.4 μsec	.9 μsec	1.6 μsec	2.5 μsec	10 μsec	1 msec
n^3	1 μsec	8 μsec	27 μsec	64 μsec	.13 msec	1 msec	1 sec
2^n	1 μsec	1 msec	1.1 sec	18.3 min	2.1 yr	2.4×10^{13} cent	----
n!	3.6 msec	1.8 yr	2.0×10^{15} cent	!	forget it	?	∞

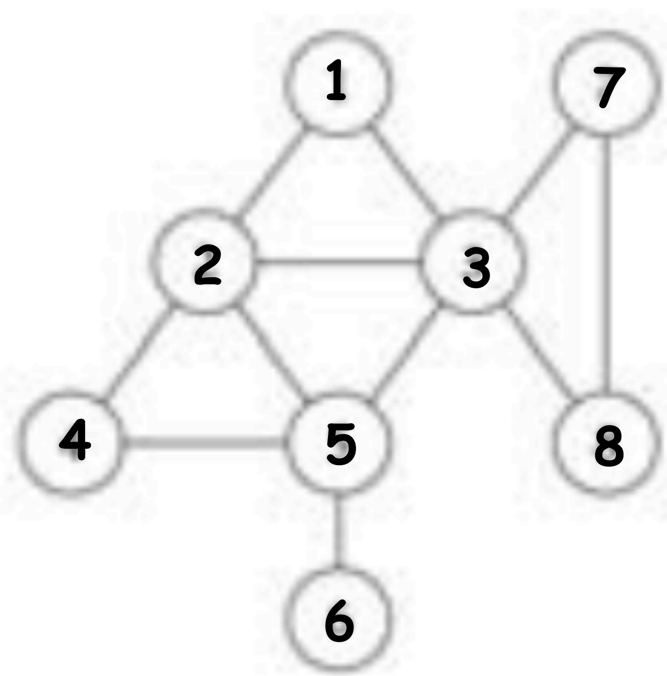
Polynomial vs. **exponential** growth

(assuming 1,000,000,000 operations per second)

Undirected Graphs

Undirected graph. $G = (V, E)$

- V = nodes.
- E = edges between pairs of nodes.
- Captures pairwise relationship between objects.
- Graph size parameters: $n = |V|$, $m = |E|$.



$V = \{ 1, 2, 3, 4, 5, 6, 7, 8 \}$

$E = \{ 1\text{-}2, 1\text{-}3, 2\text{-}3, 2\text{-}4, 2\text{-}5, 3\text{-}5, 3\text{-}7, 3\text{-}8, 4\text{-}5, 5\text{-}6 \}$

$n = 8$

$m = 11$

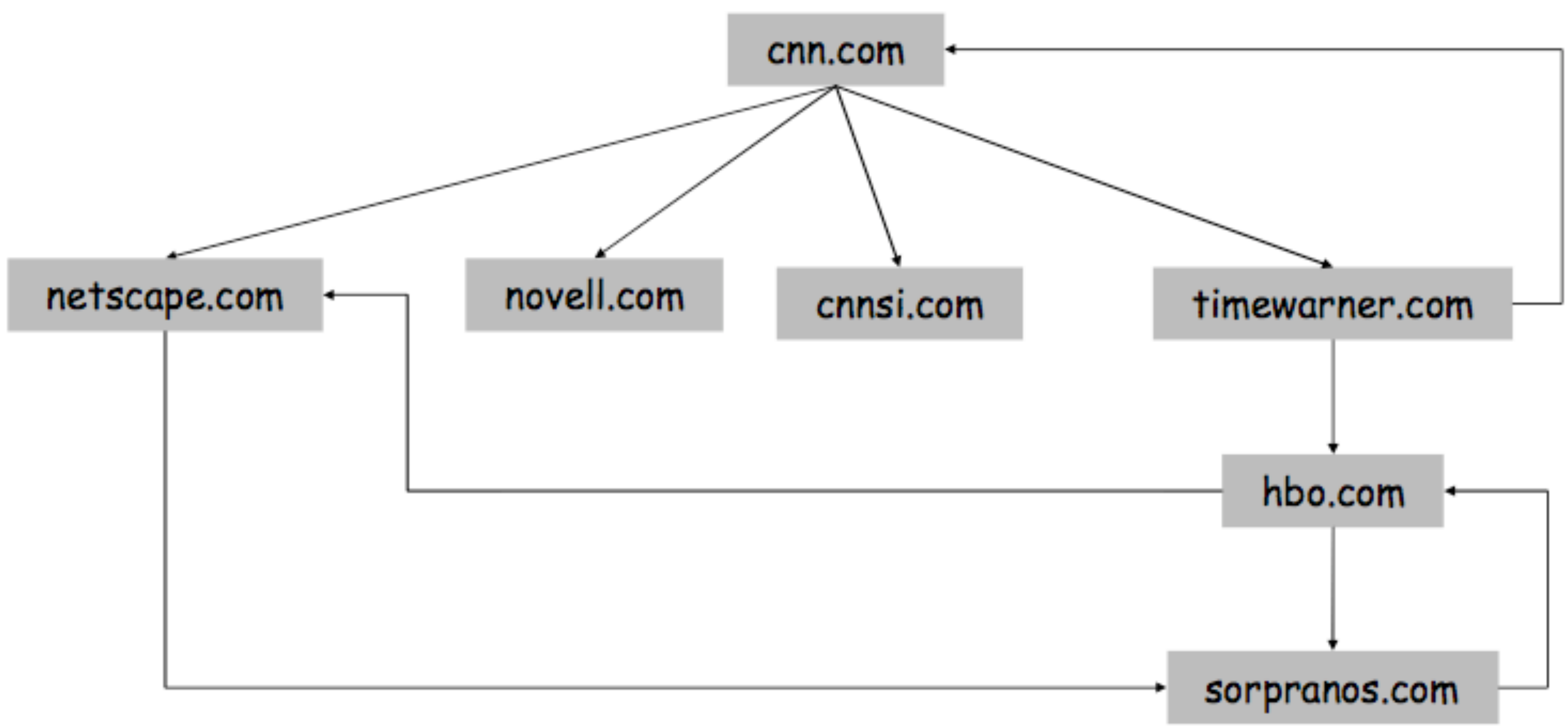
Some Graph Applications

Graph	*Nodes*	*Edges*
transportation	street intersections	highways
communication	computers	fiber optic cables
World Wide Web	web pages	hyperlinks
social	people	relationships
food web	species	predator-prey
software systems	functions	function calls
scheduling	tasks	precedence constraints
circuits	gates	wires

World Wide Web

Web graph.

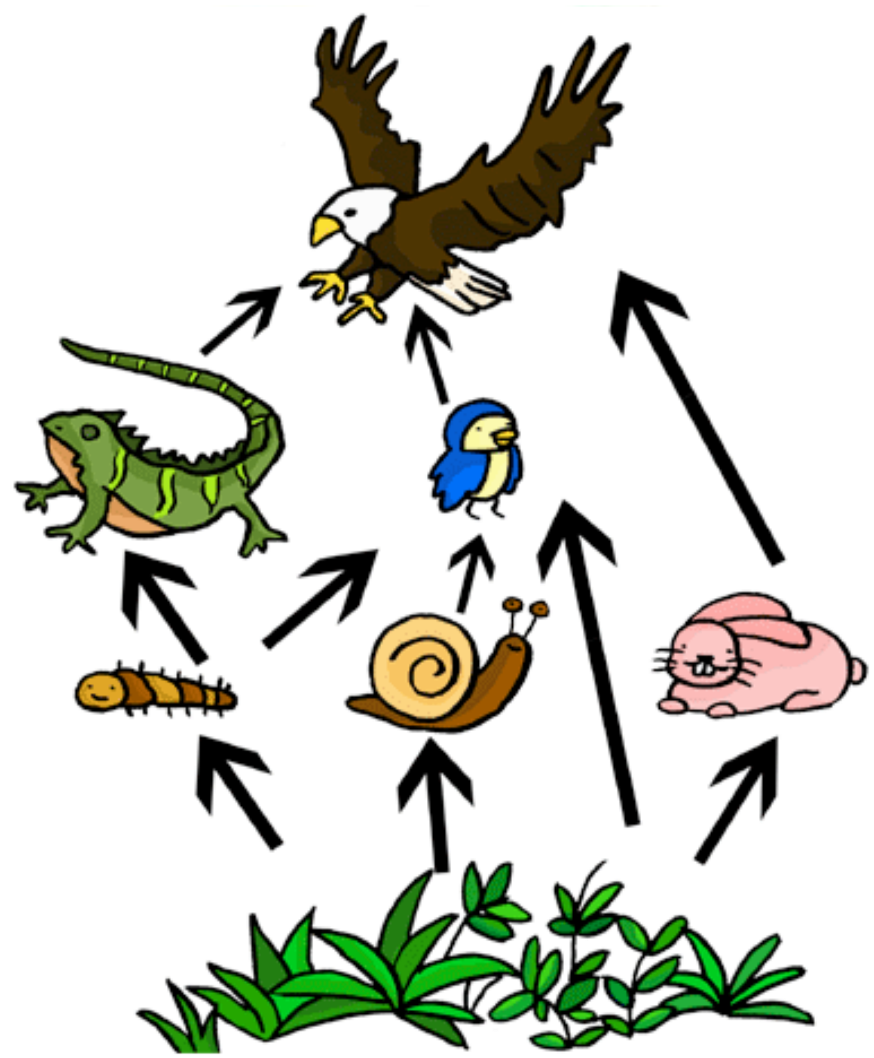
- Node: web page.
- Edge: hyperlink from one page to another.



Ecological Food Web

Food web graph.

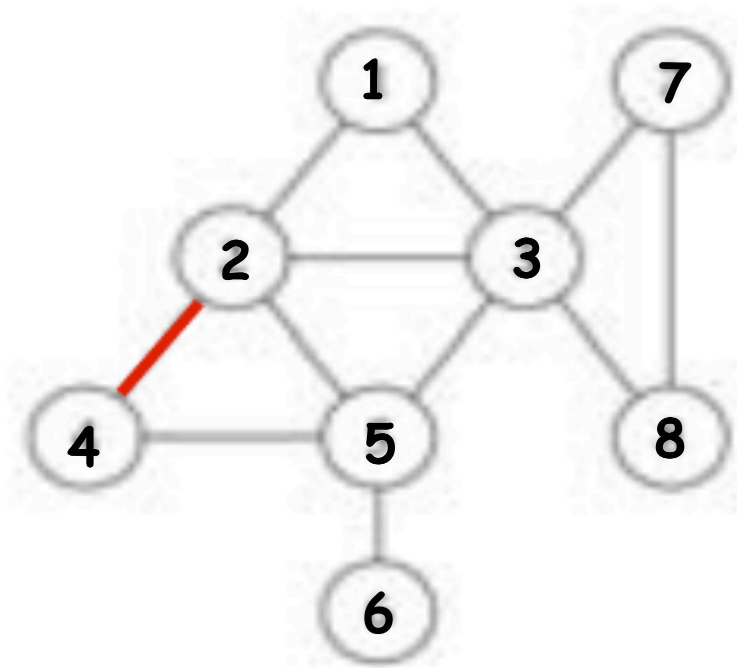
- Node = species.
- Edge = from prey to predator.



Graph Representation: Adjacency Matrix

Adjacency matrix. n-by-n matrix with $A_{uv} = 1$ if (u, v) is an edge.

- Two representations of each edge.
- Space proportional to n^2.
- Checking if (u, v) is an edge takes $\Theta(1)$ time.
- Identifying all edges takes $\Theta(n^2)$ time.



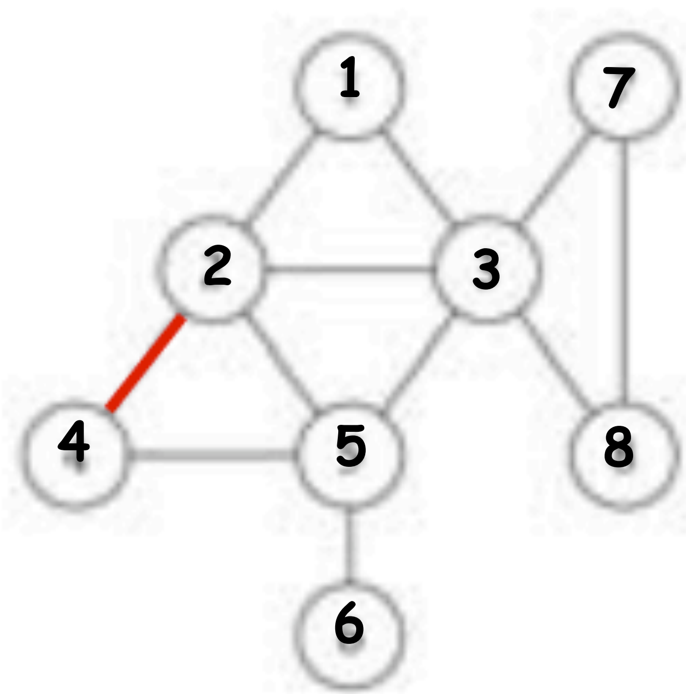
	1	2	3	4	5	6	7	8
1	0	1	1	0	0	0	0	0
2	1	0	1	1	1	0	0	0
3	1	1	0	0	1	0	1	1
4	0	1	0	0	1	0	0	0
5	0	1	1	1	0	1	0	0
6	0	0	0	0	1	0	0	0
7	0	0	1	0	0	0	0	1
8	0	0	1	0	0	0	1	0

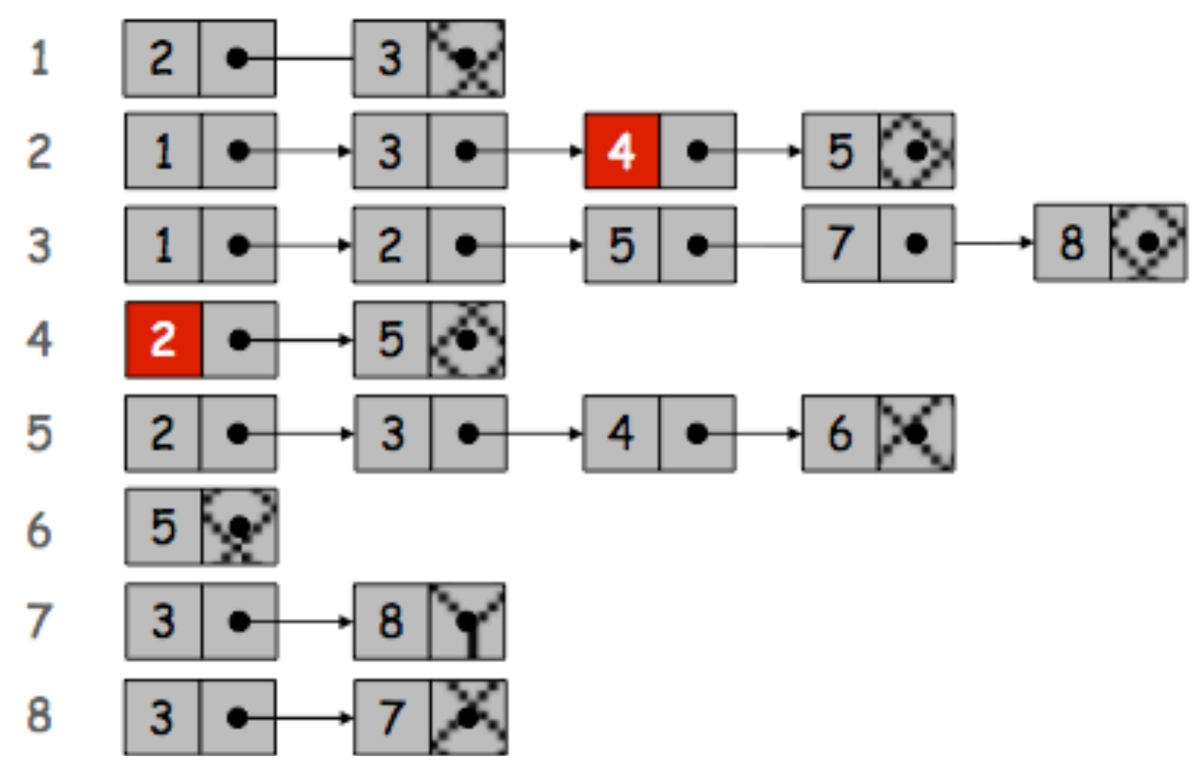
Graph Representation: Adjacency List

Adjacency list. Node indexed array of lists.

- Two representations of each edge.
- Space proportional to m + n.
- Checking if (u, v) is an edge takes O(deg(u)) time.
- Identifying all edges takes Θ(m + n) time.

degree = number of neighbors of u



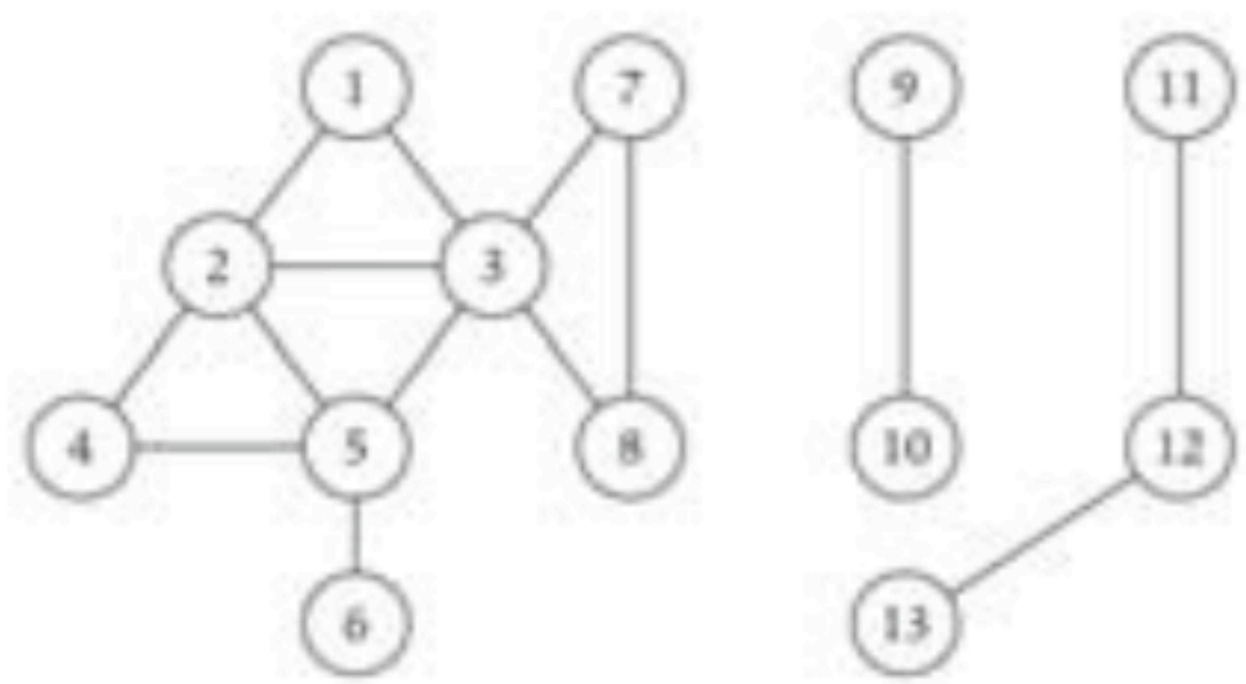


Paths and Connectivity

Def. A path in an undirected graph $G = (V, E)$ is a sequence P of nodes $v_1, v_2, \ldots, v_{k-1}, v_k$ with the property that each consecutive pair v_i, v_{i+1} is joined by an edge in E.

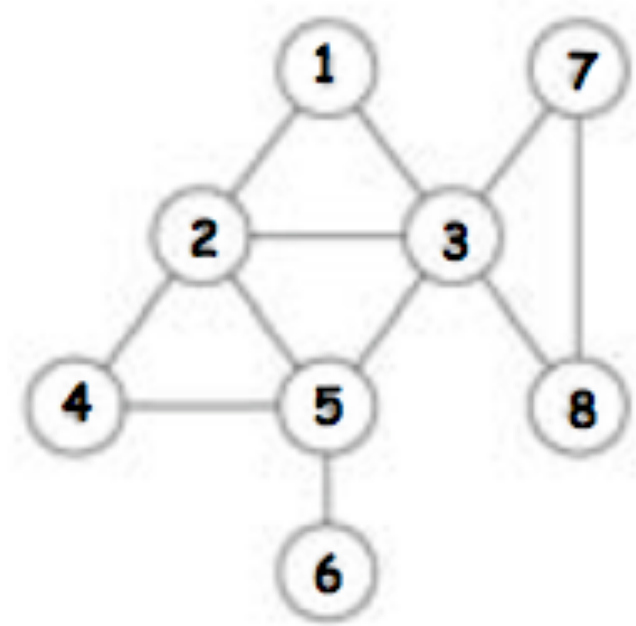
Def. A path is simple if all nodes are distinct.

Def. An undirected graph is connected if for every pair of nodes u and v, there is a path between u and v.



Cycles

Def. A cycle is a path $v_1, v_2, \ldots, v_{k-1}, v_k$ in which $v_1 = v_k$, $k > 2$, and th
first k-1 nodes are all distinct.



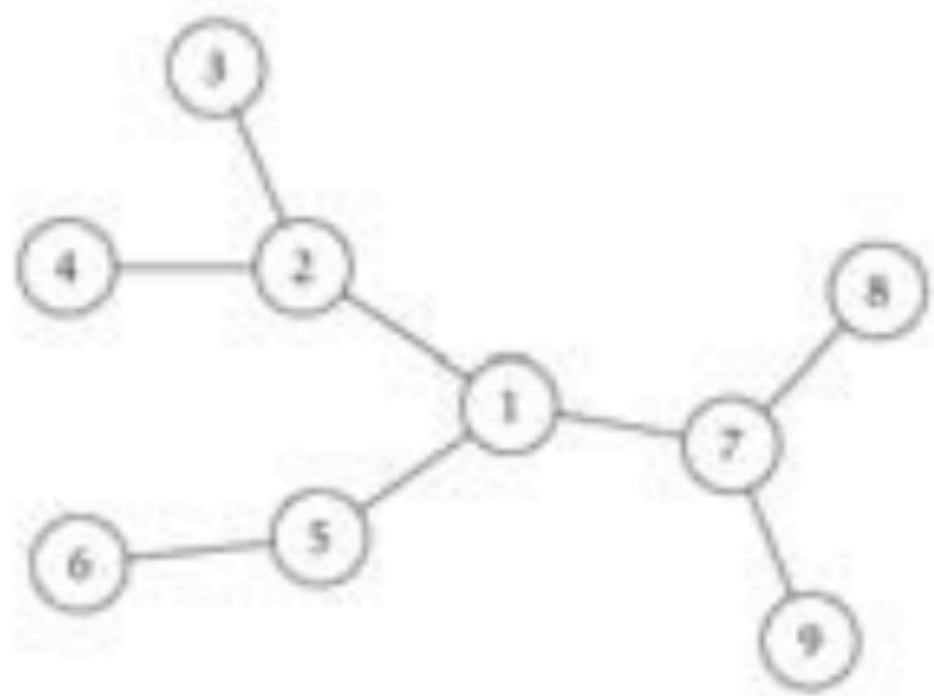
cycle C = 1-2-4-5-3-1

Trees

Def. An undirected graph is a tree if it is connected and does not contain a cycle.

Theorem. Let G be an undirected graph on n nodes. Any two of the following statements imply the third.

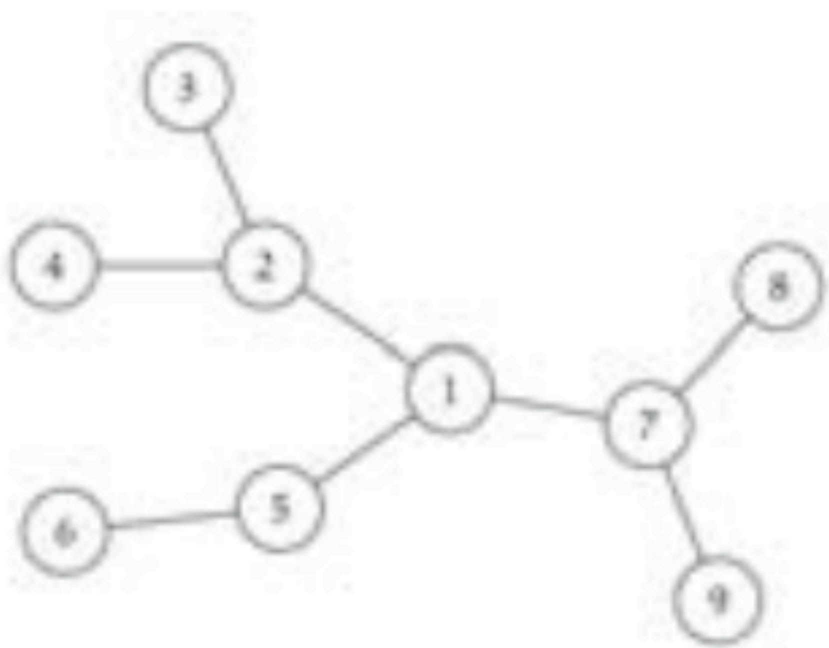
- G is connected.
- G does not contain a cycle.
- G has n-1 edges.



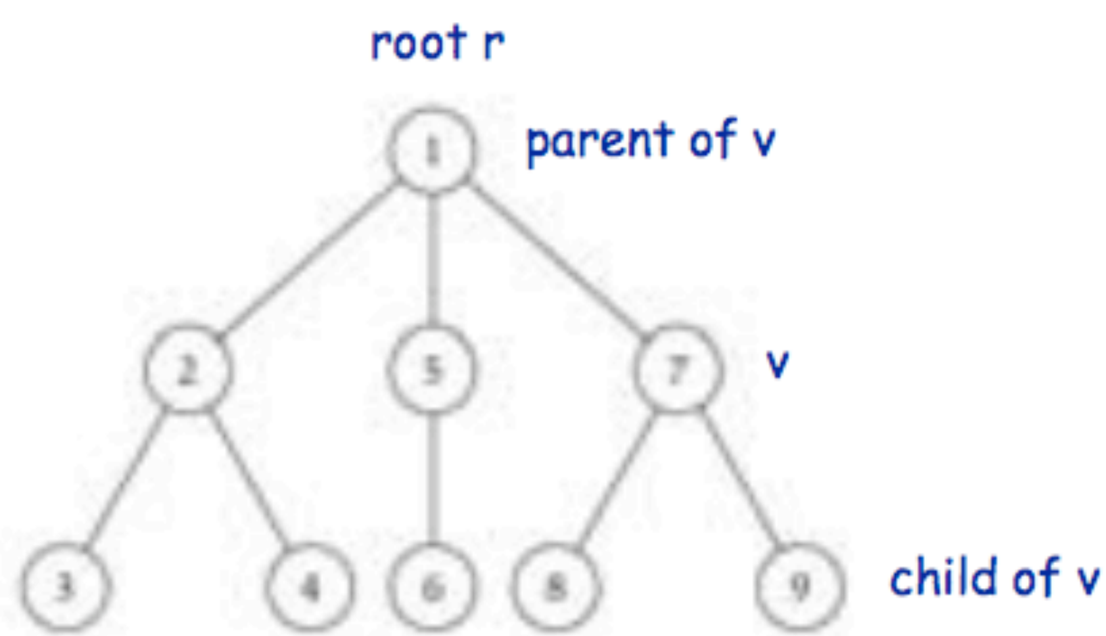
Rooted Trees

Rooted tree. Given a tree T, choose a root node r and orient each edge away from r.

Importance. Models hierarchical structure.



a tree



the same tree, rooted at 1

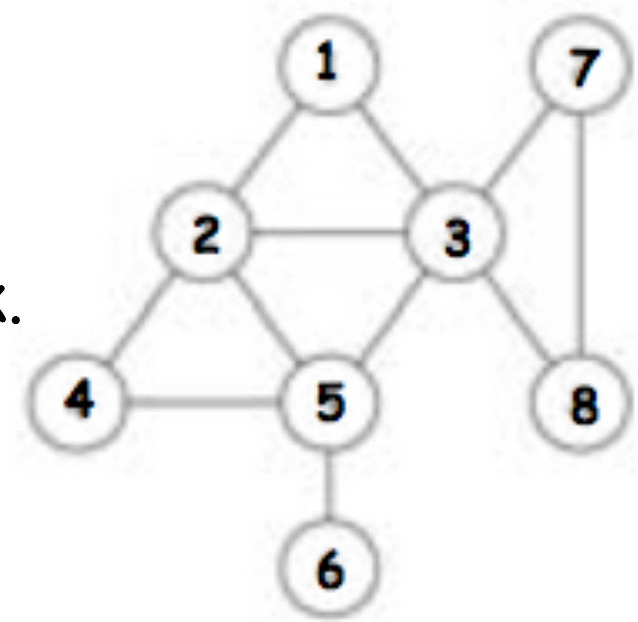
Connectivity

s-t connectivity problem. Given two node s and t, is there a path between s and t?

s-t shortest path problem. Given two node s and t, what is the length of the shortest path between s and t?

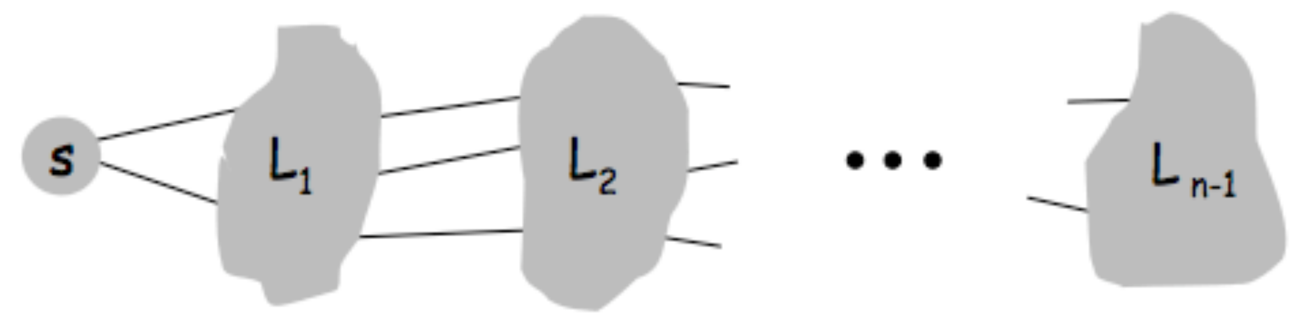
Applications.

- Friendster.
- Maze traversal.
- Kevin Bacon number.
- Fewest number of hops in a communication network.



Breadth First Search

BFS intuition. Explore outward from s in all possible directions, adding nodes one "layer" at a time.



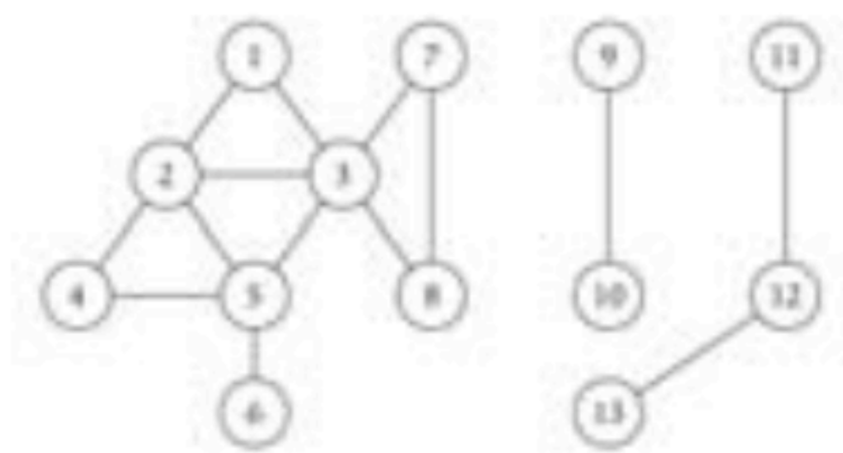
BFS algorithm.

- $L_0 = \{ s \}$.
- L_1 = all neighbors of L_0.
- L_2 = all nodes that do not belong to L_0 or L_1, and that have an edge to a node in L_1.
- L_{i+1} = all nodes that do not belong to an earlier layer, and that have an edge to a node in L_i.

Theorem. For each i, L_i consists of all nodes at distance exactly i from s. There is a path from s to t iff t appears in some layer.

Connected Component

Connected component. Find all nodes reachable from s.



Connected component containing node 1 = { 1, 2, 3, 4, 5, 6, 7, 8 }.

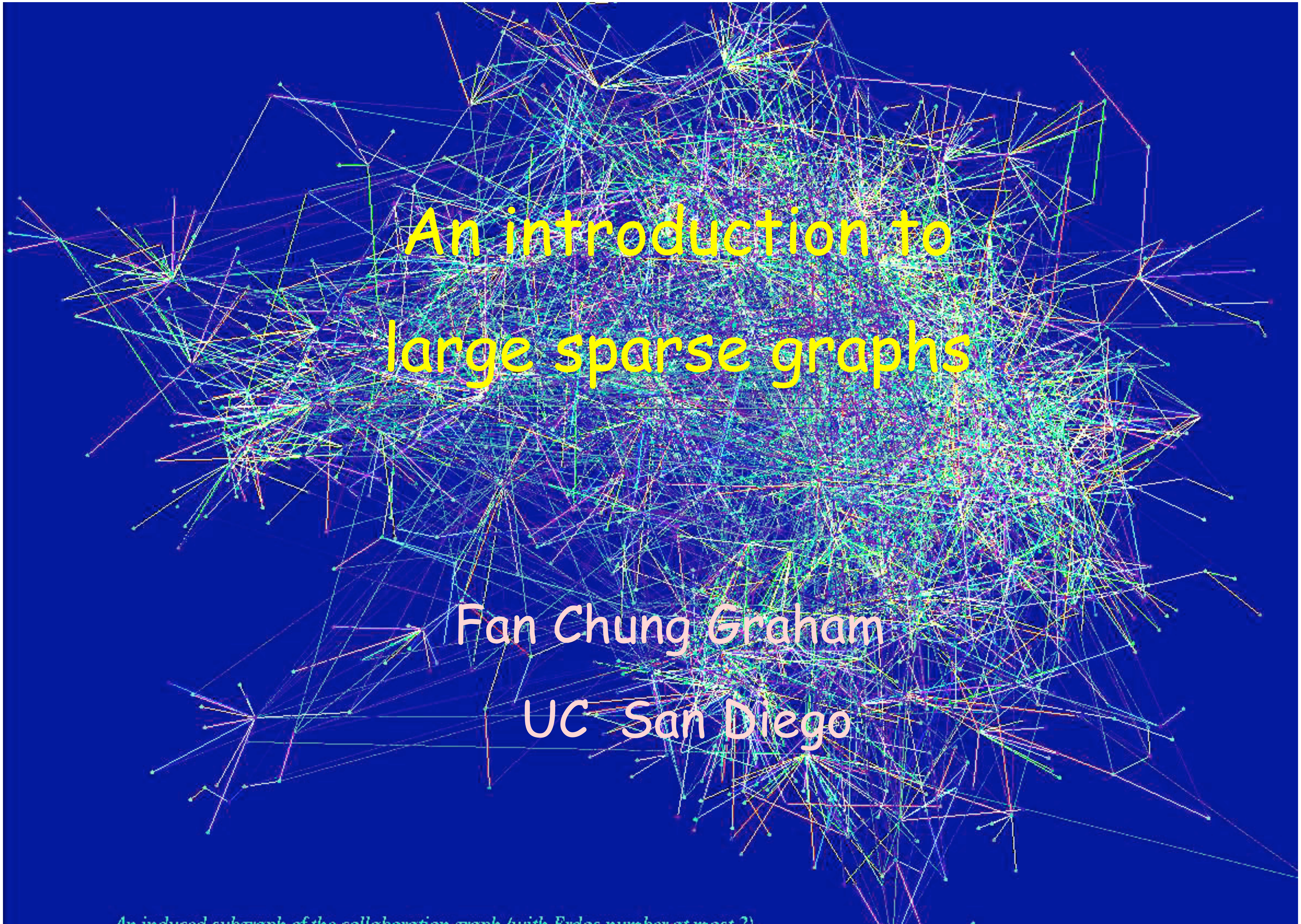
Algorithms: Breadth First Search BFS

Depth First Search DFS

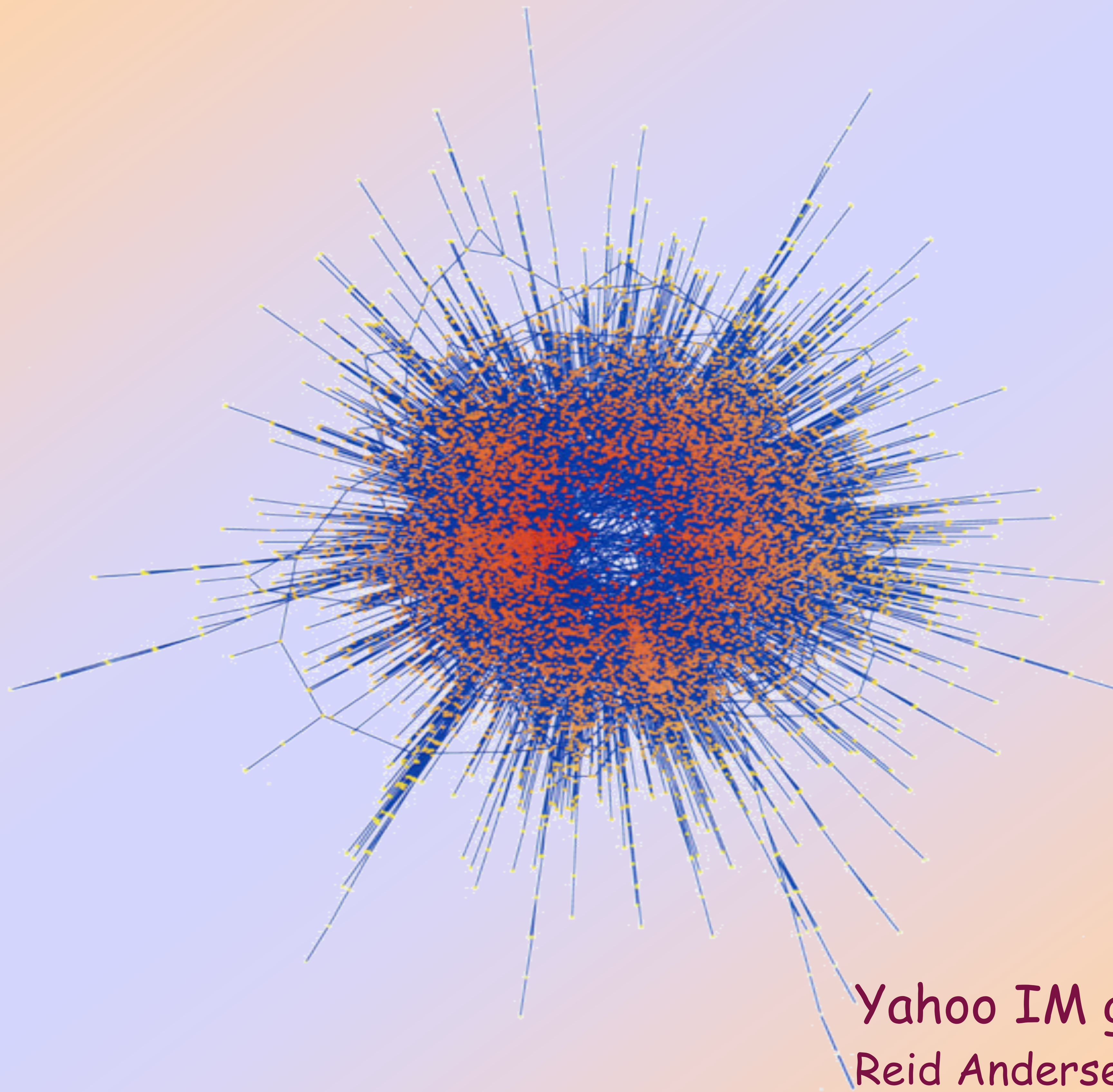
An introduction to large sparse graphs

Fan Chung Graham

UC San Diego



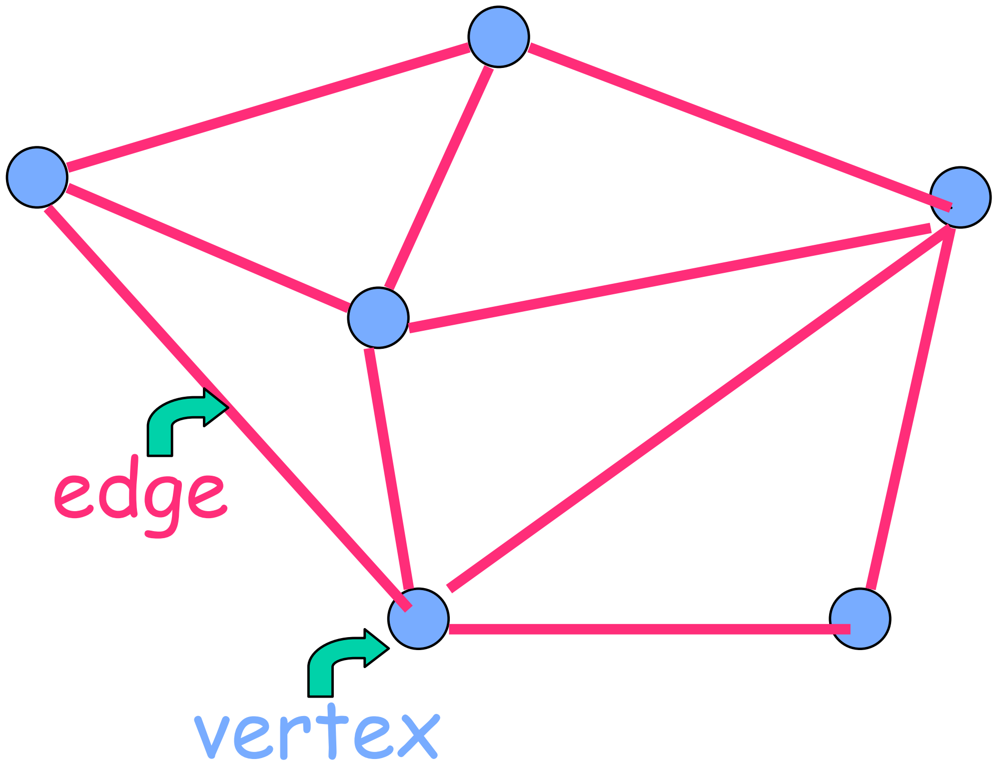
An induced subgraph of the collaboration graph (with Erdos number at most 2).
Made by Fan Chung Graham and Lincoln Lu in 2002.



Yahoo IM graph

Reid Andersen 2005

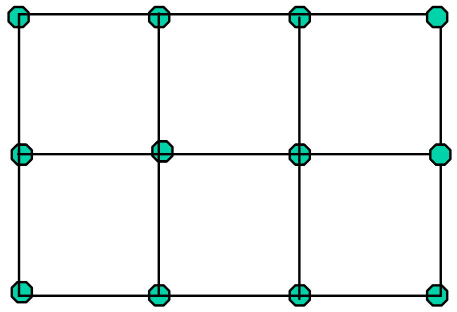
A graph G = (V,E)

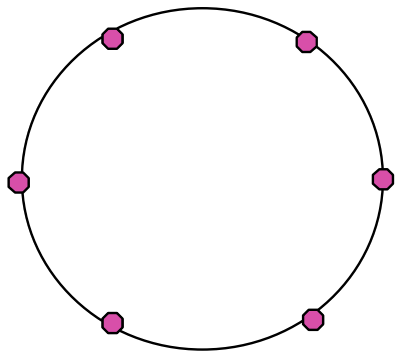


Graph Theory has 250 years of history.

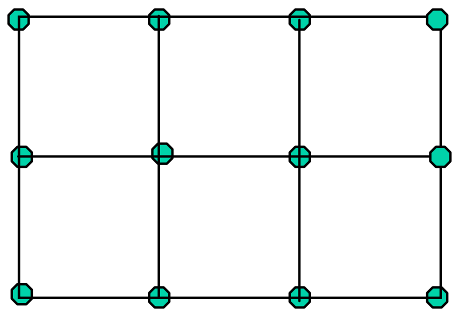
Leonhard Euler, 1707-1783

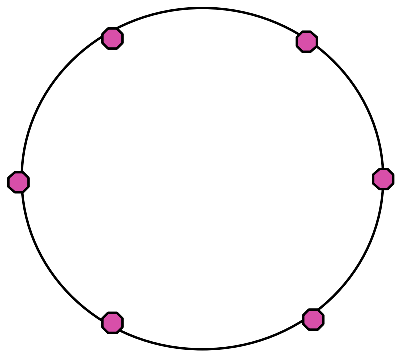
Geometric graphs

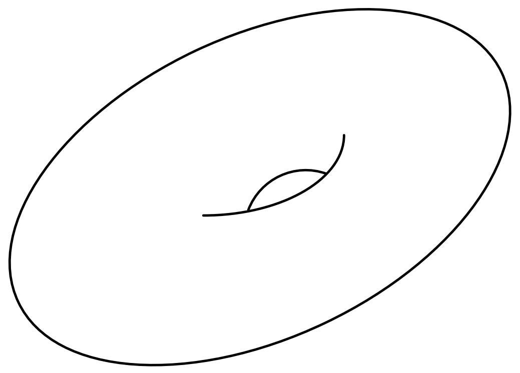




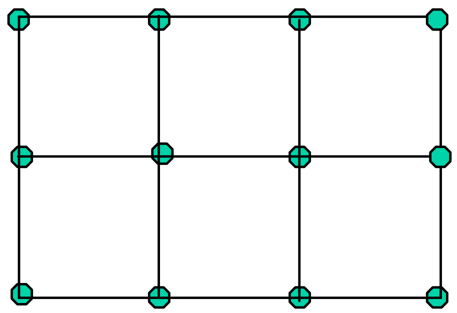
Geometric graphs

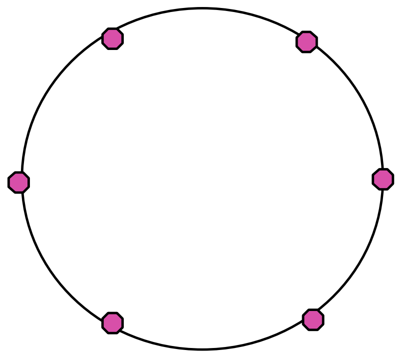


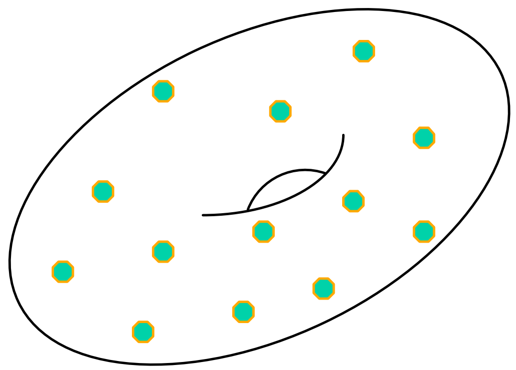




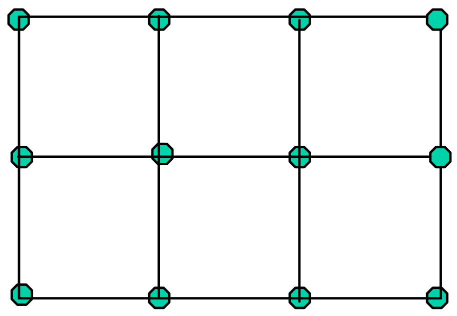
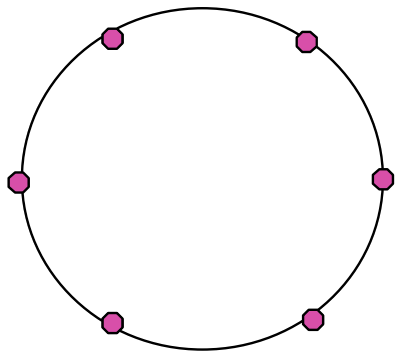
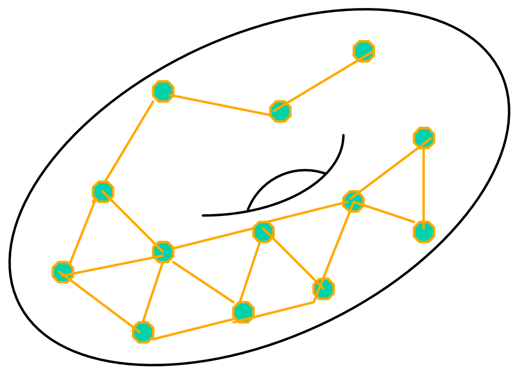
Geometric graphs



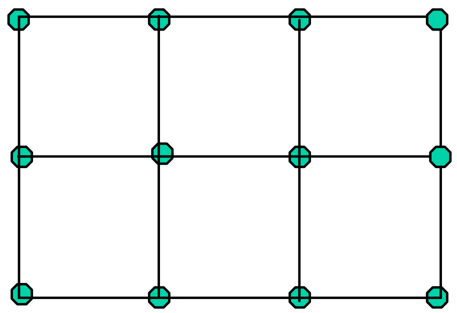


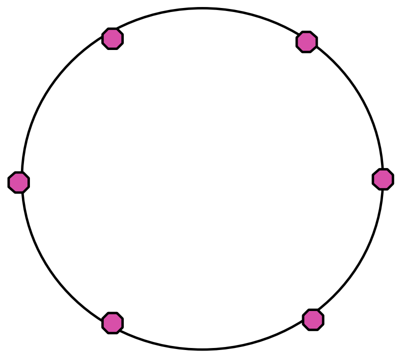


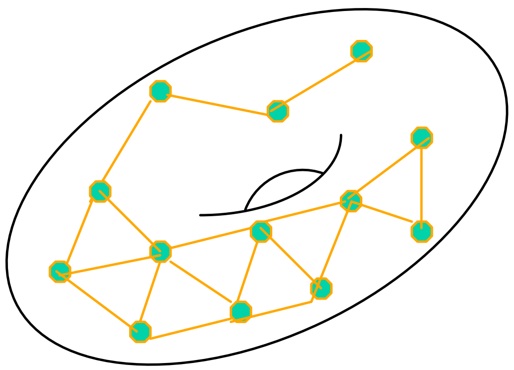
Geometric graphs

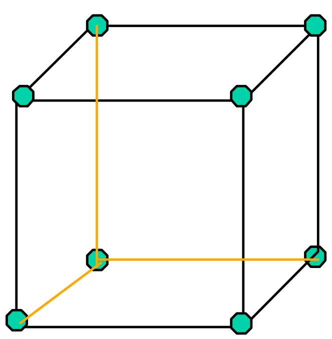




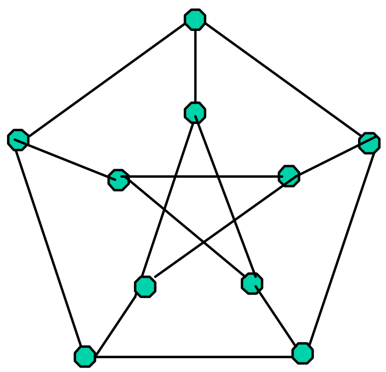
Geometric graphs





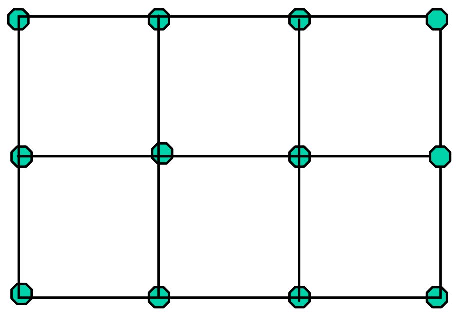


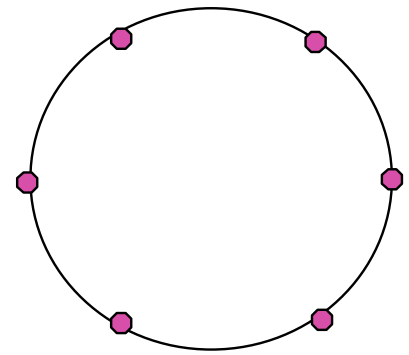


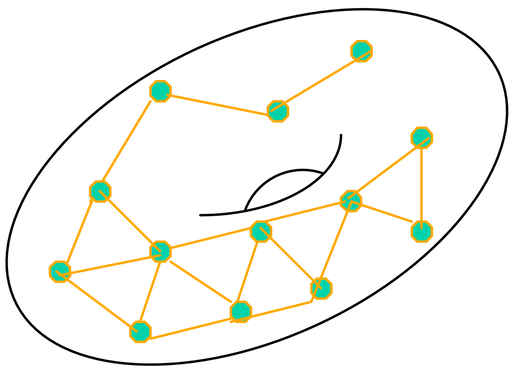


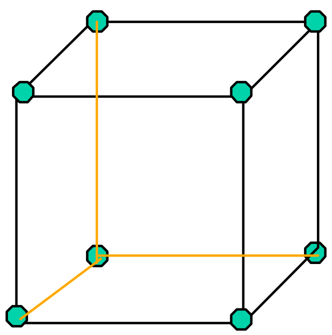
Algebraic graphs

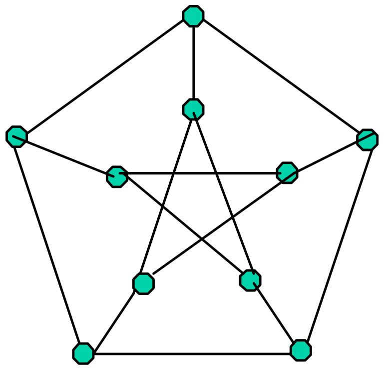
Geometric graphs



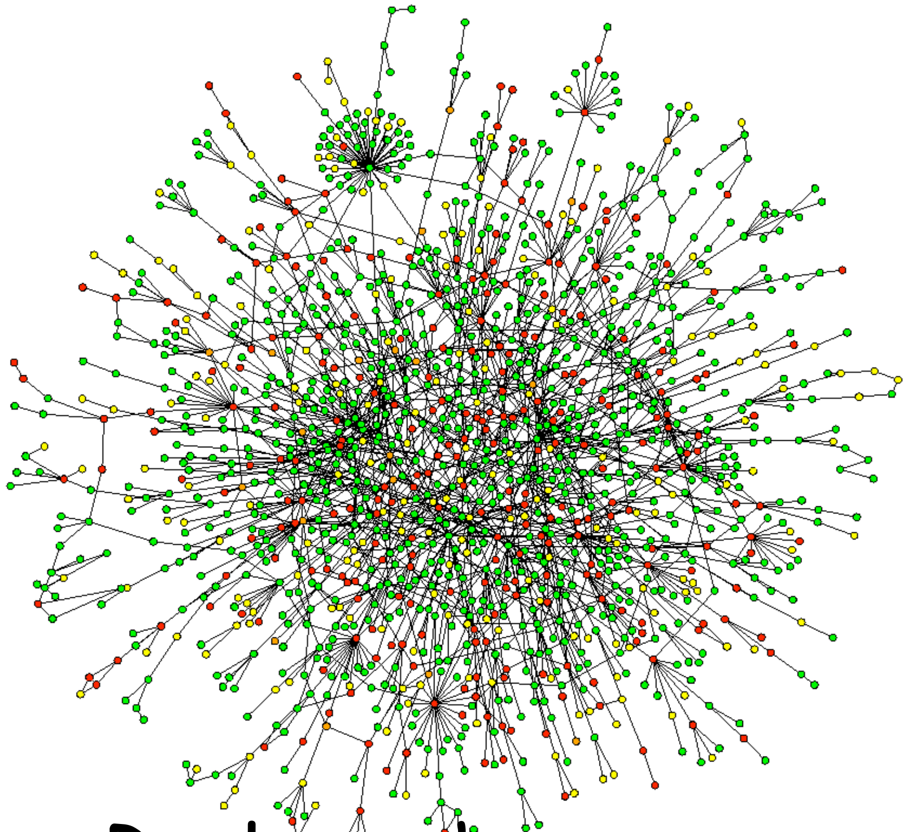




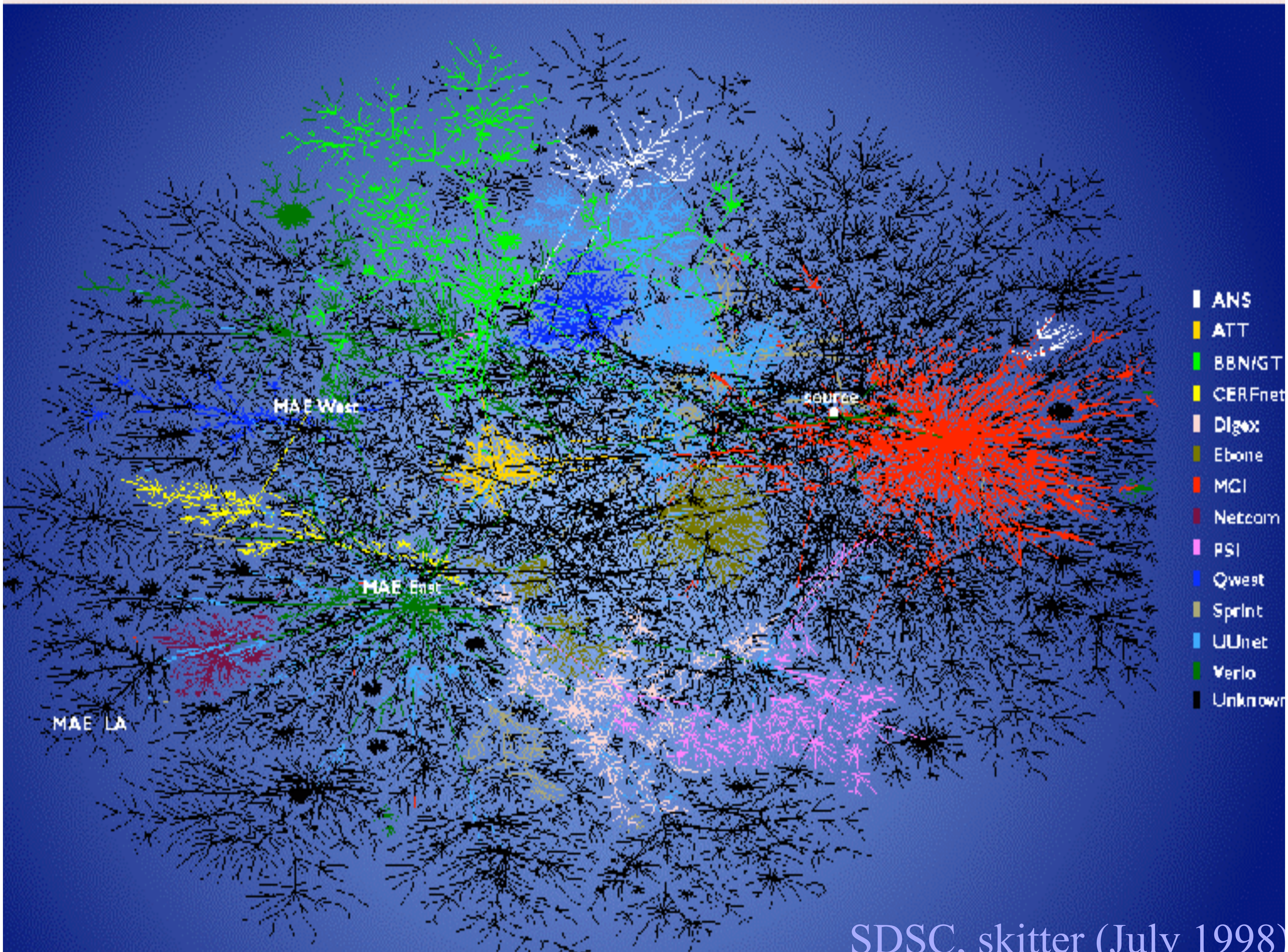




Algebraic graphs



Real graphs
(protein interactions by Jawoong Jeong)

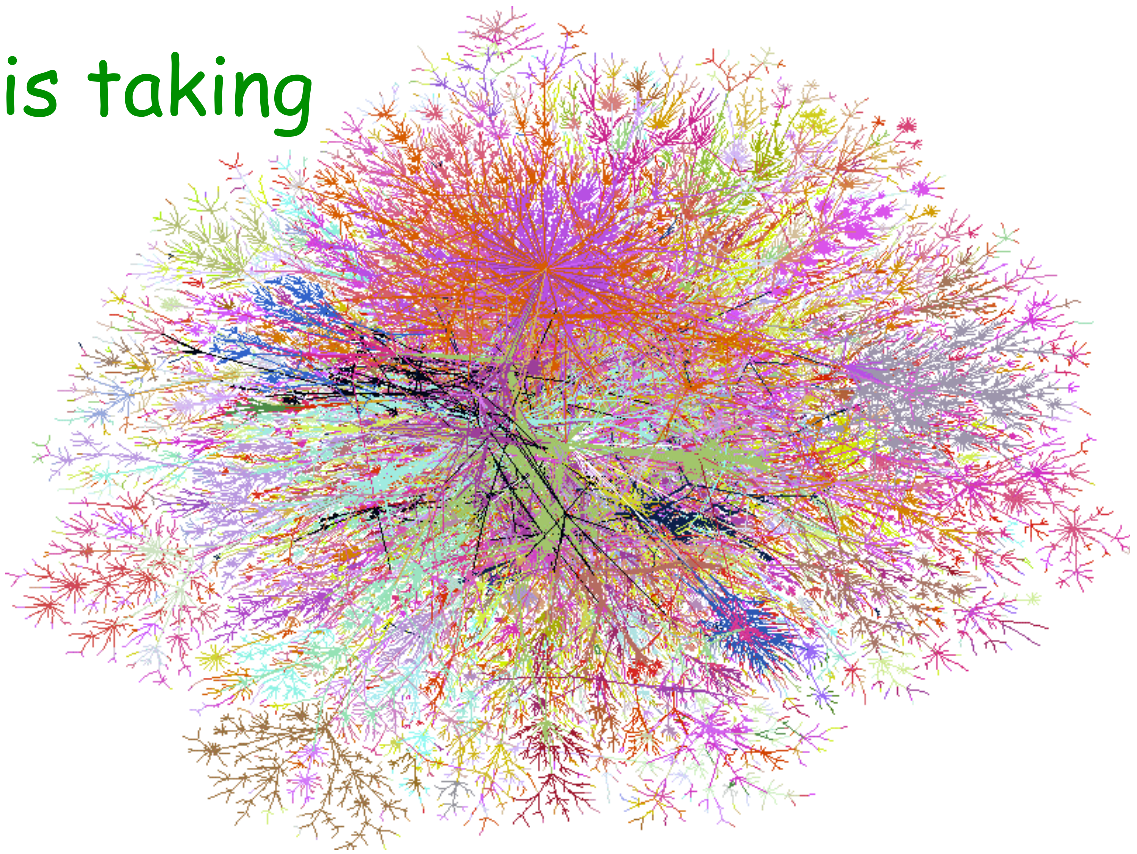

MAE West
source
MAE East
MAE LA
ANS
ATT
BBN/GT
CERFnet
Digex
Ebone
MCI
Netcom
PSI
Qwest
Sprint
UUnet
Verio
Unknown
SDSC, skitter (July 1998)

Massive data

Massive graphs

Massive data → Massive graphs

The information we deal with is taking on a networked character.



What does a massive graph look like?

What does a massive graph look like?

- sparse
- clustered
- small diameter

What does a massive graph look like?

sparse

clustered

small diameter

prohibitively large

dynamically changing

incomplete information

What does a massive graph look like?

sparse

clustered

small diameter

prohibitively large

dynamically changing

incomplete information

Hard to describe !

Harder to analyze !!

Some prevailing characteristic of large realistic networks

- Small world phenomenon

  Small diameter/average distance

  Clustering

- Power law degree distribution

A crucial observation

Massive graphs satisfy the power law.

Discovered by several groups independently.

- Barabási, Albert and Jeung, 1999.
- Broder, Kleinberg, Kumar, Raghavan, Rajagopalan and Tomkins, 1999.
- M Faloutsos, P. Faloutsos and C. Faloutsos, 1999.
- Abello, Buchsbaum, Reeds and Westbrook, 1999.
- Aiello, Chung and Lu, 1999.

The history of power law

- Zipf's law, 1949. (The n^{th} most frequent word occurs at rate 1/n)
- Yule's law, 1942. (City population follows a power law.)
- Lotka's law, 1926. (distribution of authors in chemical abstracts) 1907-1916
- Pareto, 1897 (Wealth distribution follows a power law.)

Natural language

Bibliometrics

Social sciences

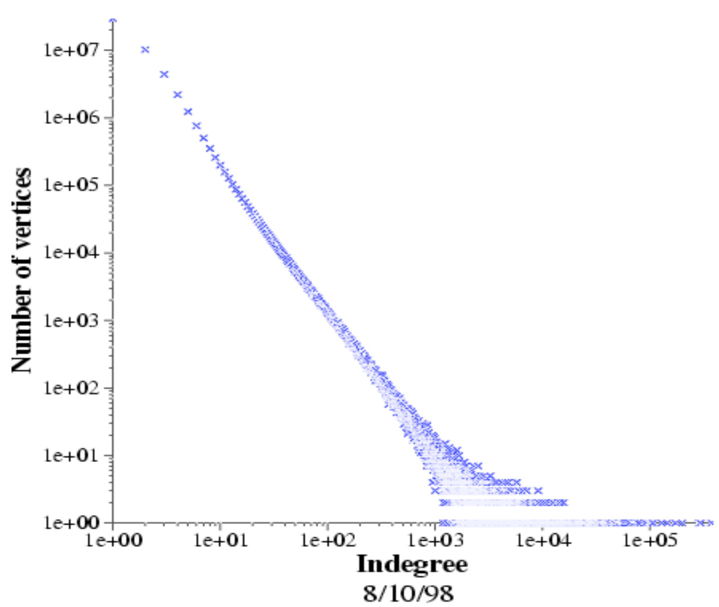
Nature

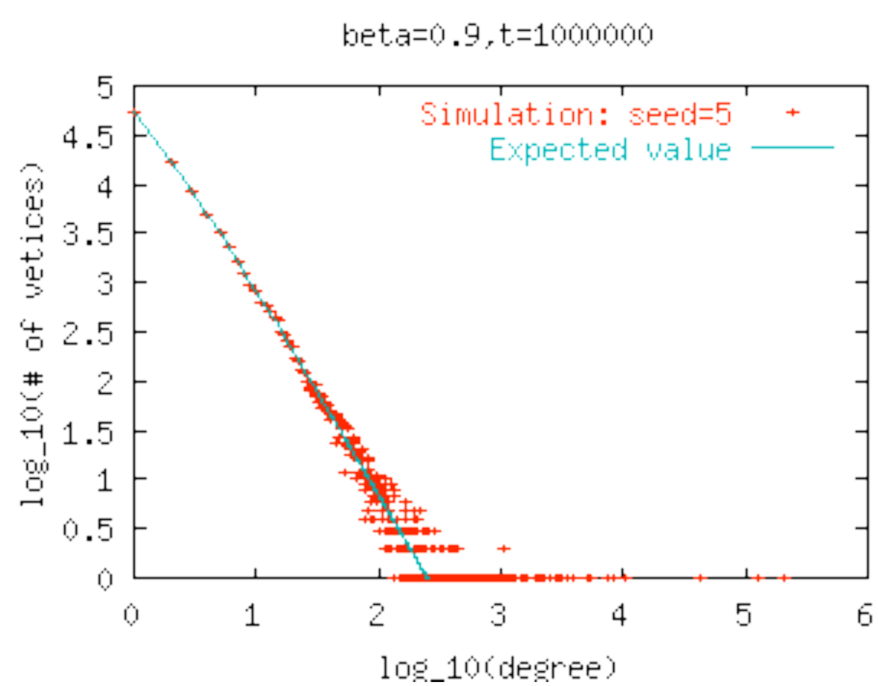
Massive graphs satisfy the power law.

Power decay degree distribution.

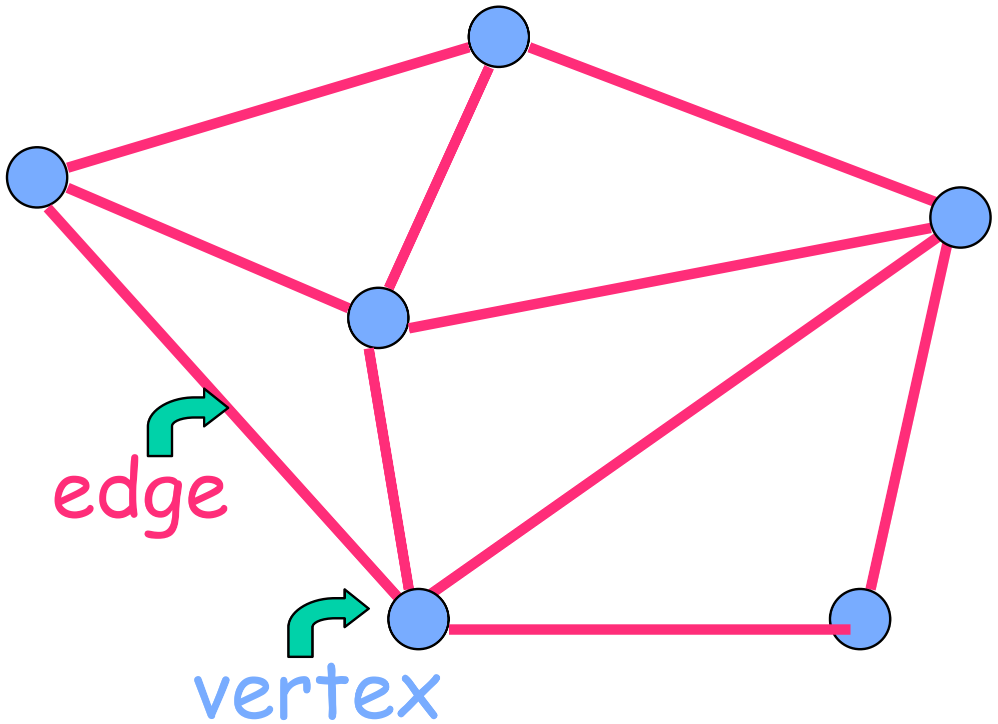
The degree sequences satisfy the power law:

The number of vertices of degree j is proportional to $j^{-\beta}$ where β is some constant ≥ 2.



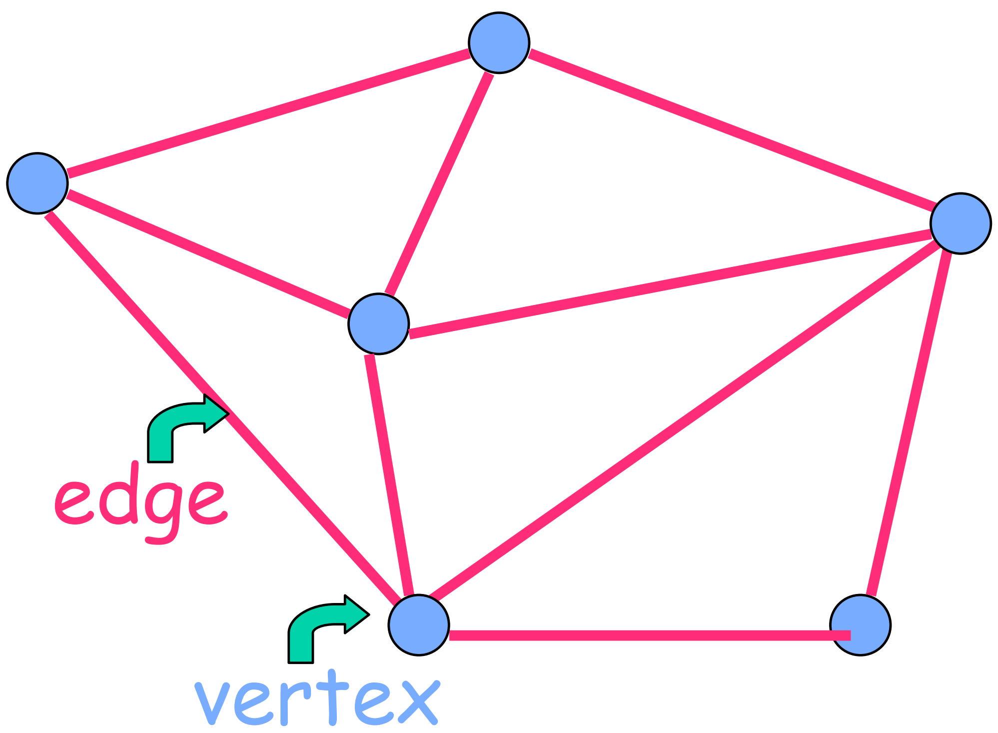


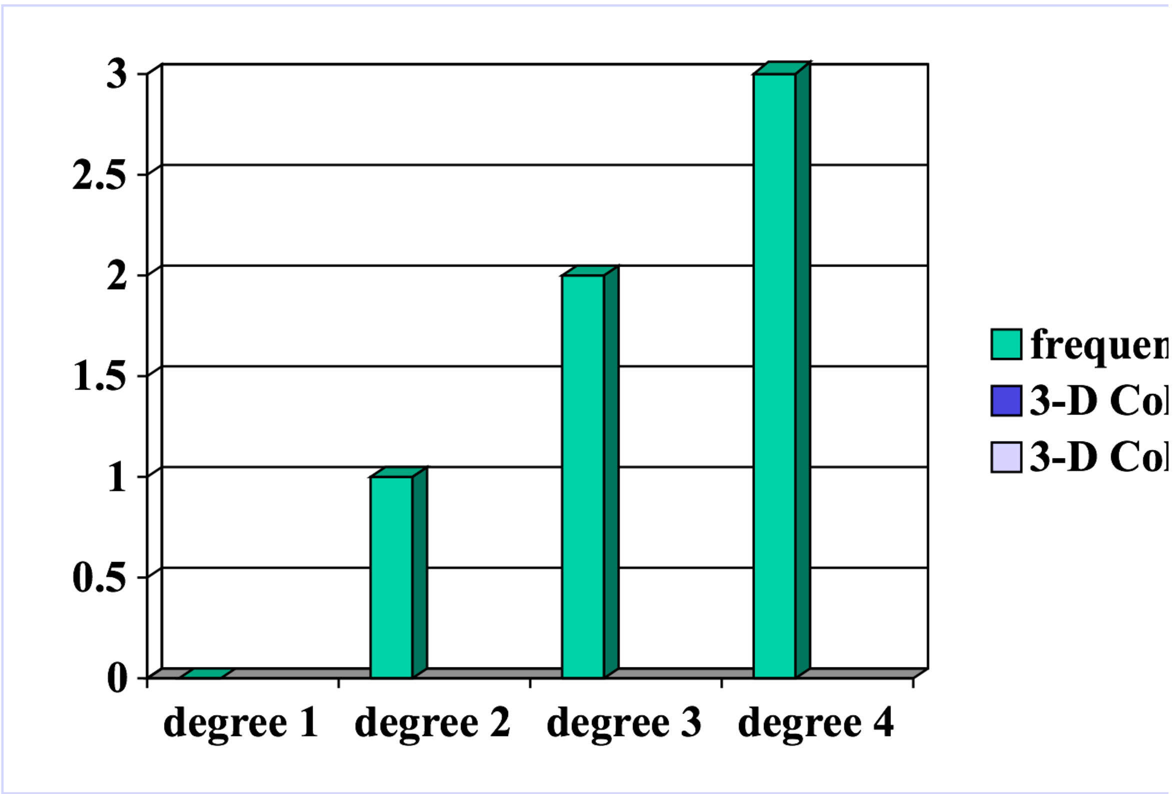
A graph G = (V,E)



Degree sequence $(4,4,4,3,3,2)=(d_i)$, d_i: **degree of** v_i

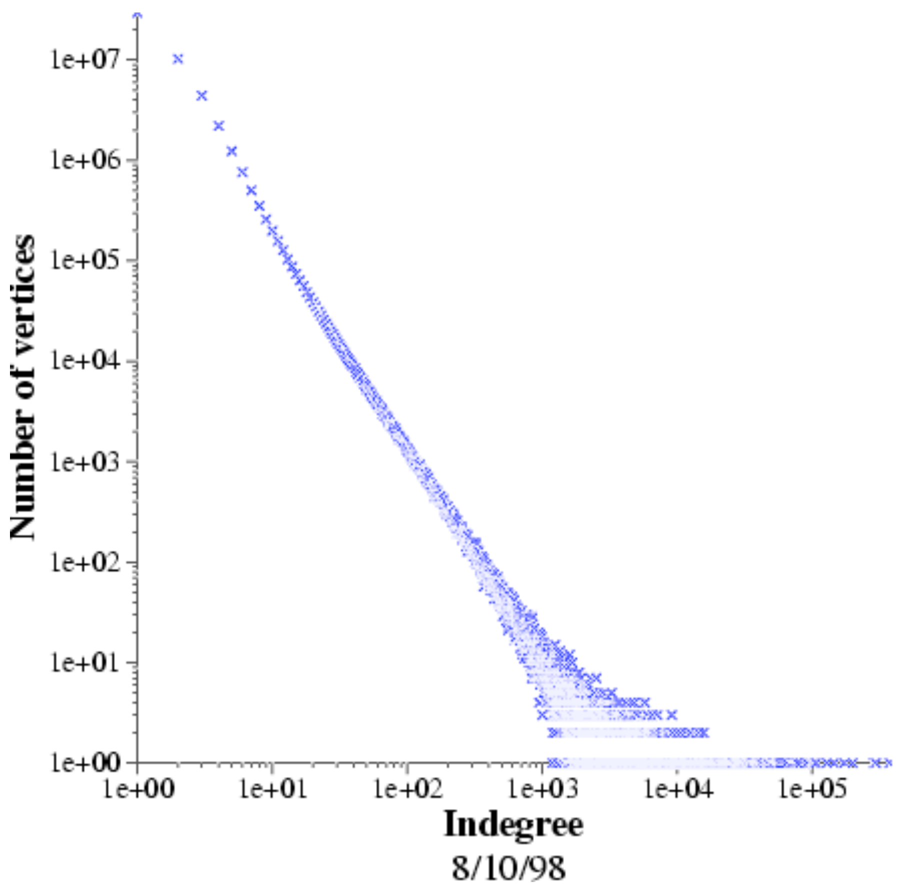
Degree distribution $(0,0,1,2,3)=(f_i)$,
f_i: **no. of vertices with degree i.**

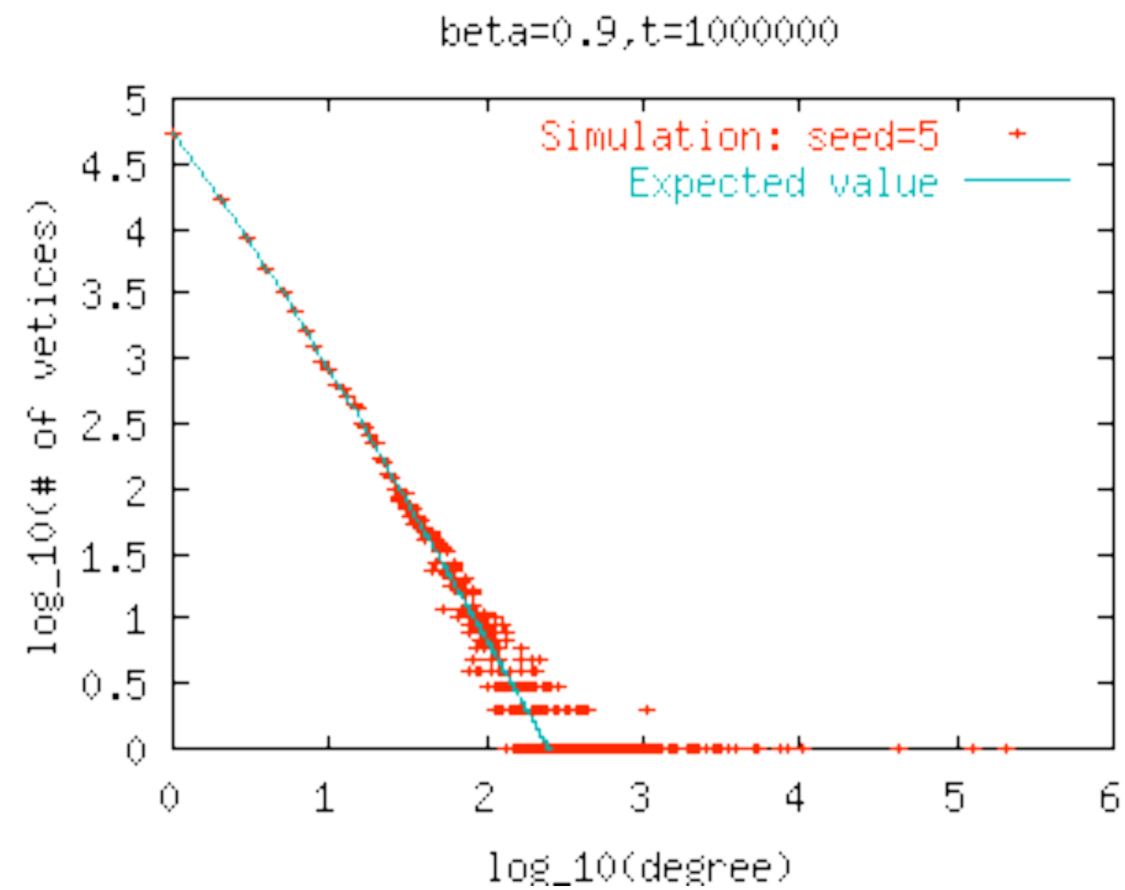




Degree distribution (0,0,1,2,3)=(f_i),
f_i: no. of vertices with degree i.

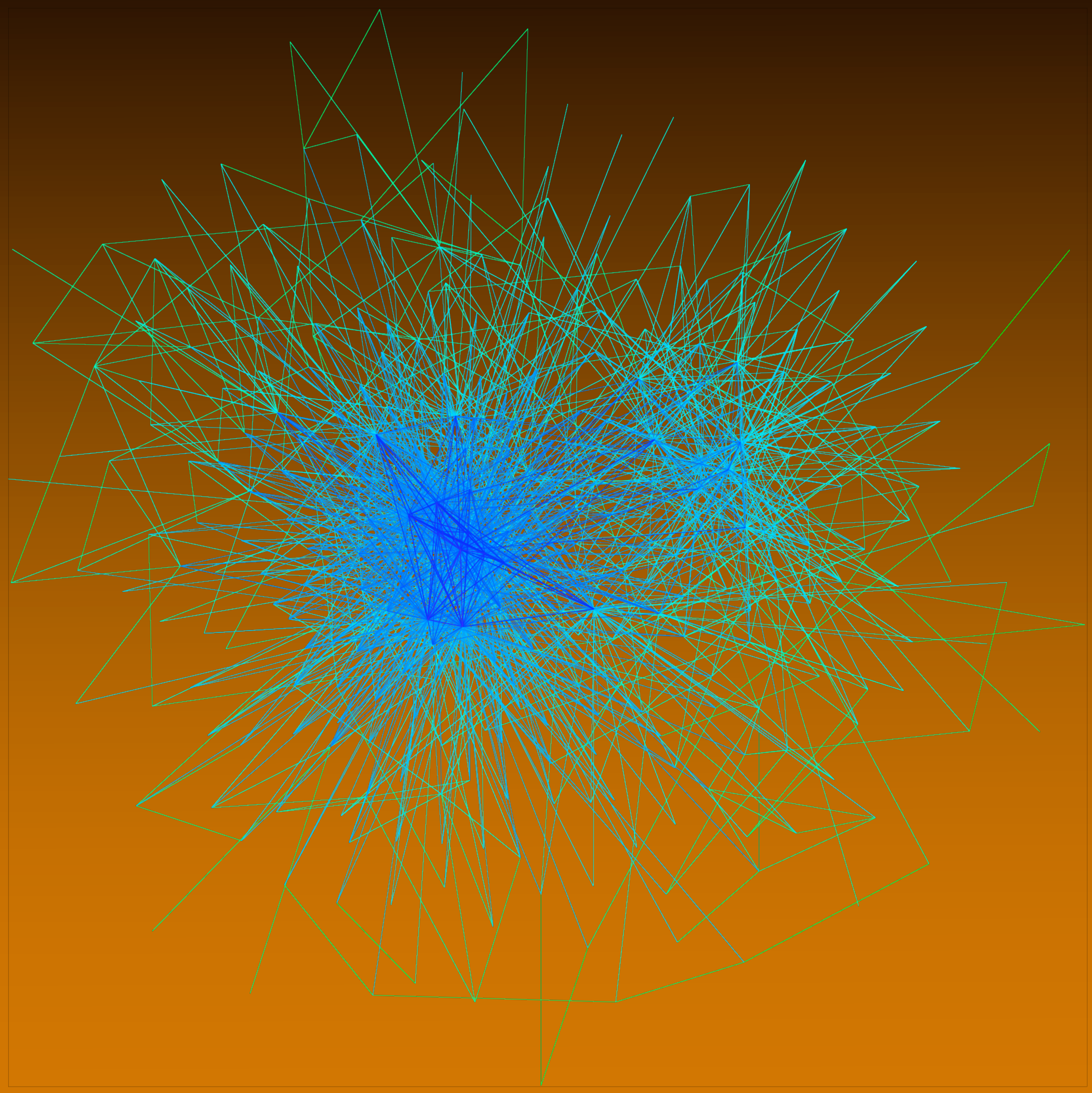
Comparisons



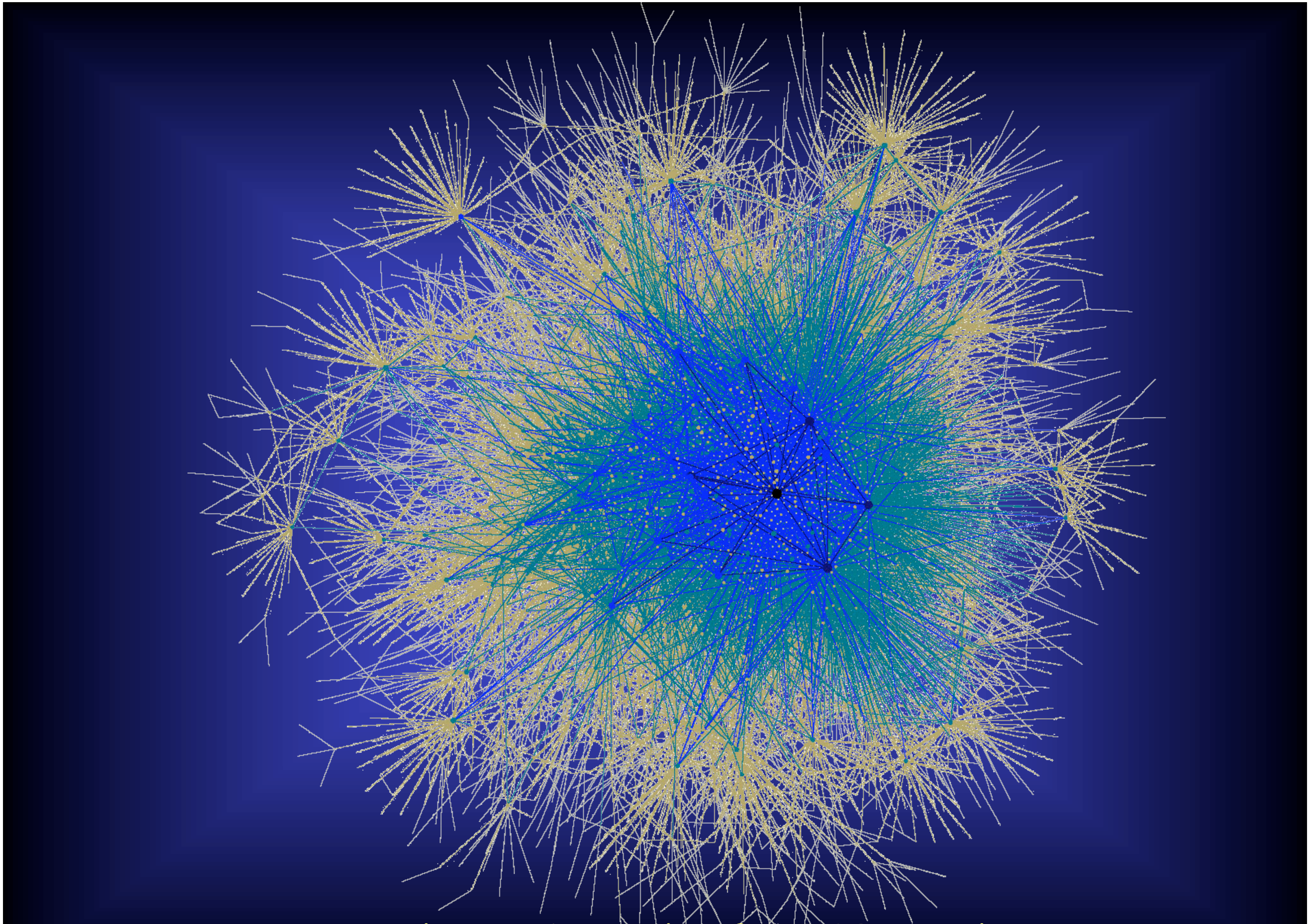


From real data

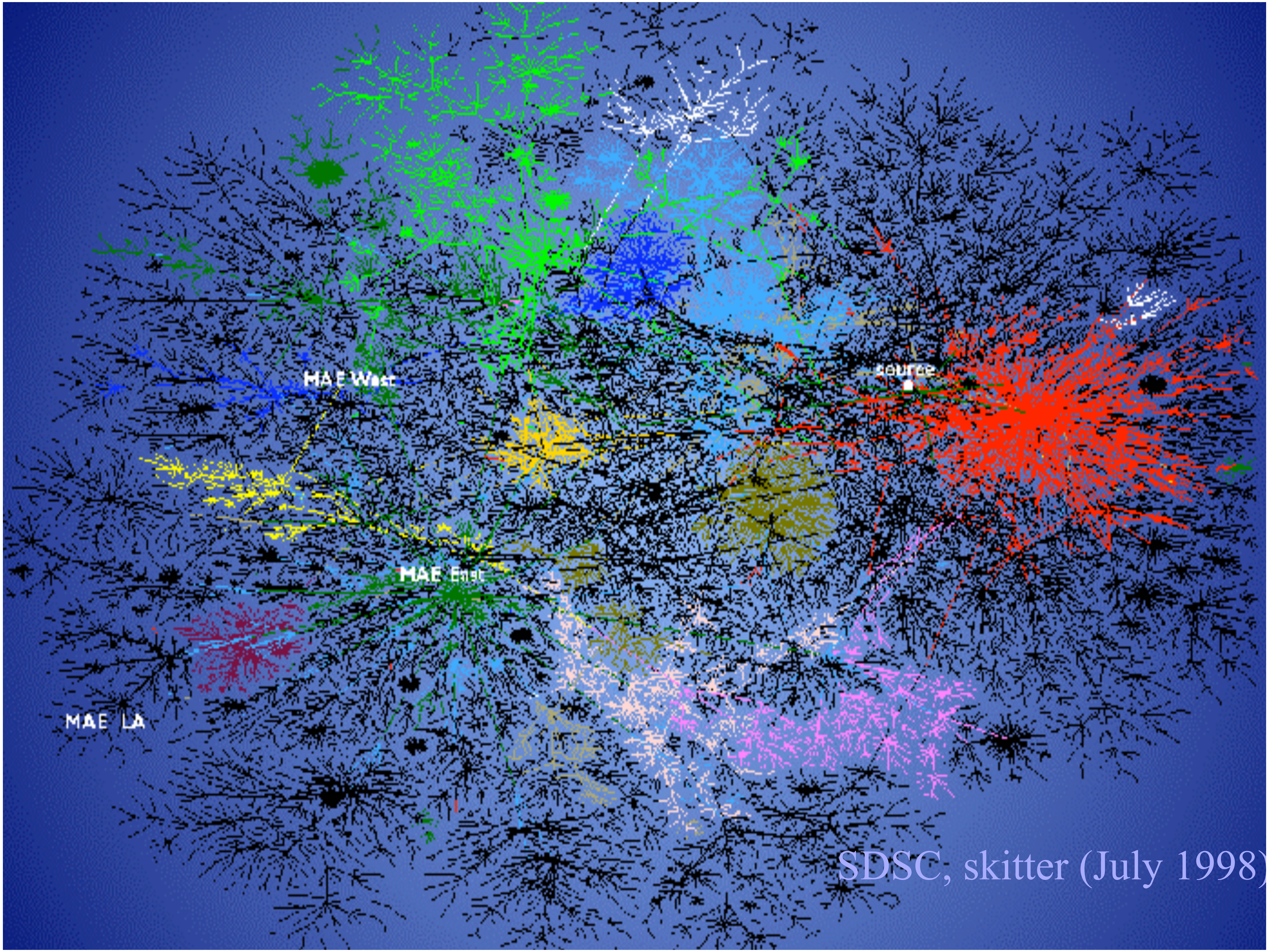
From simulation

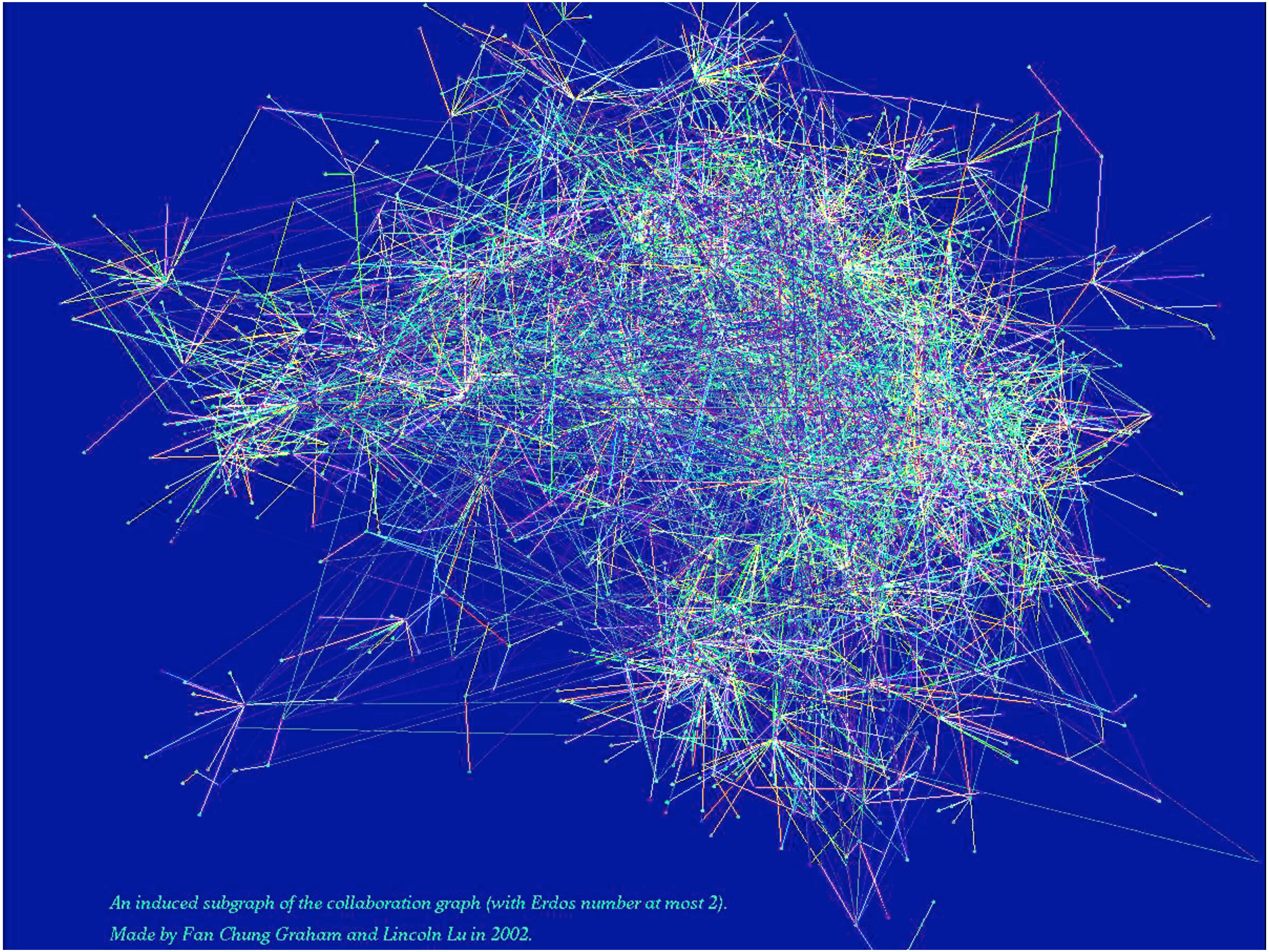


A subgraph of a BGP graph



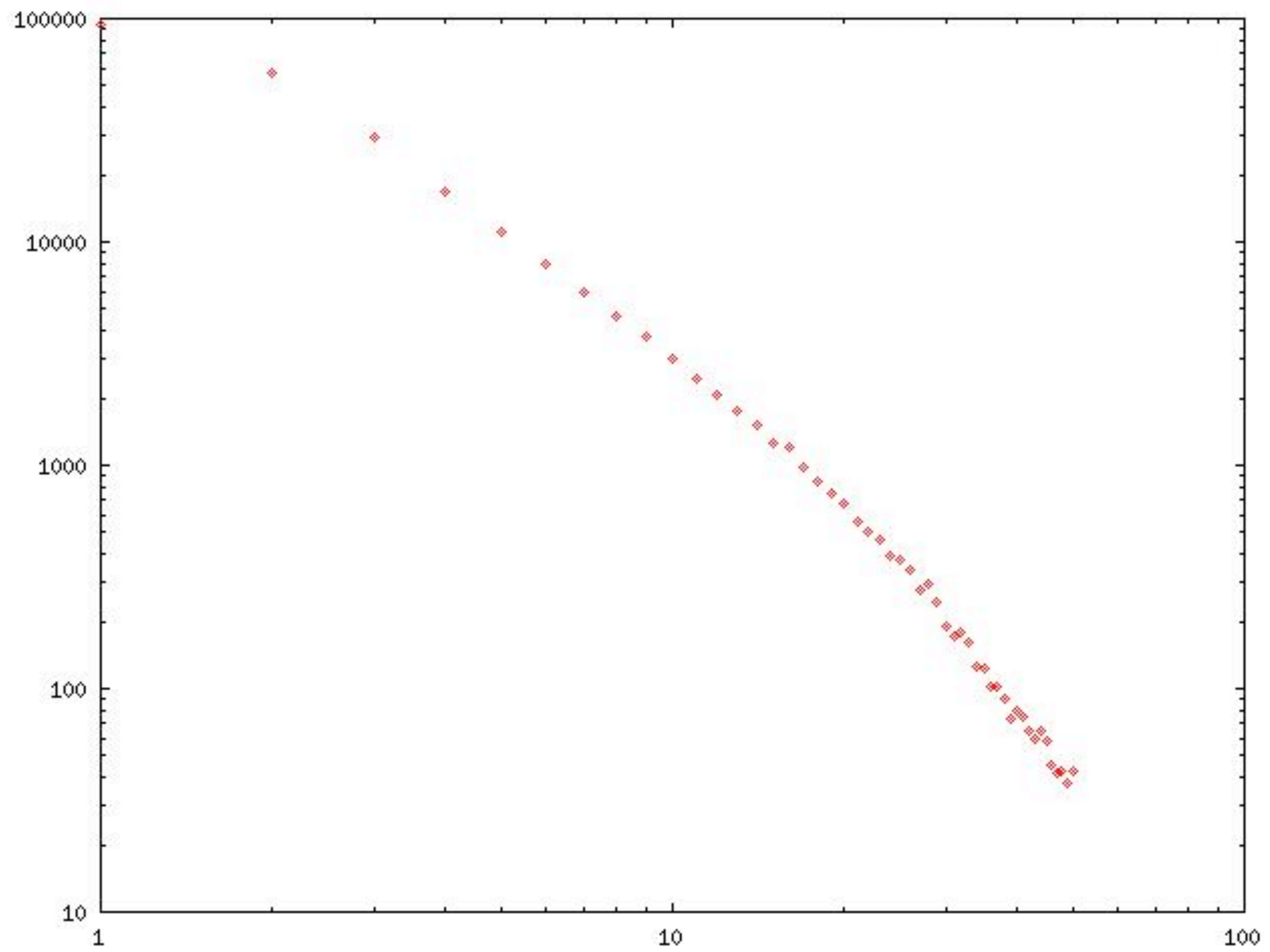
Another subgraph of a BGP graph


MAE West
source
MAE East
MAE LA
SDSC, skitter (July 1998)



An induced subgraph of the collaboration graph (with Erdos number at most 2).

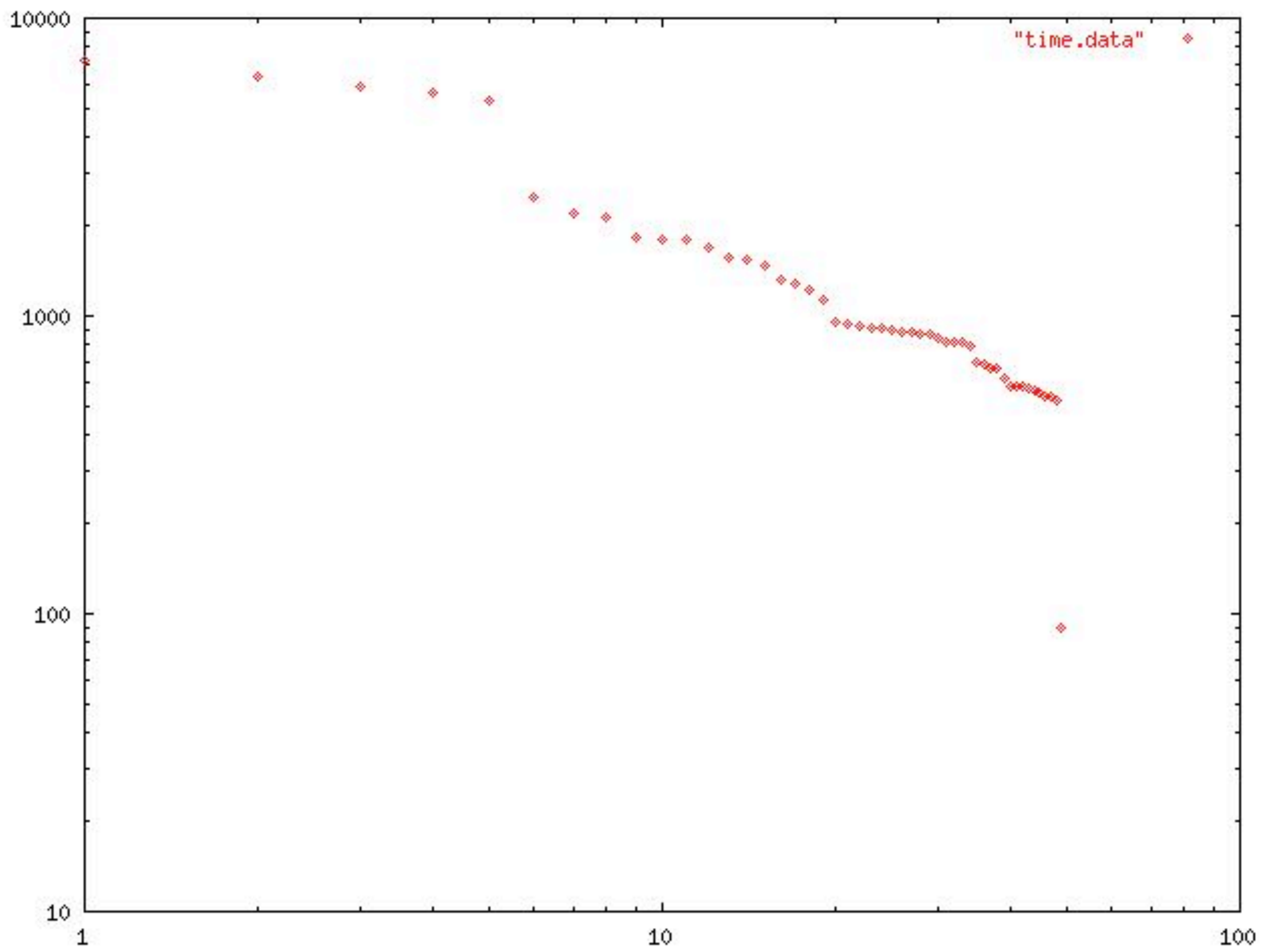
Made by Fan Chung Graham and Lincoln Lu in 2002.



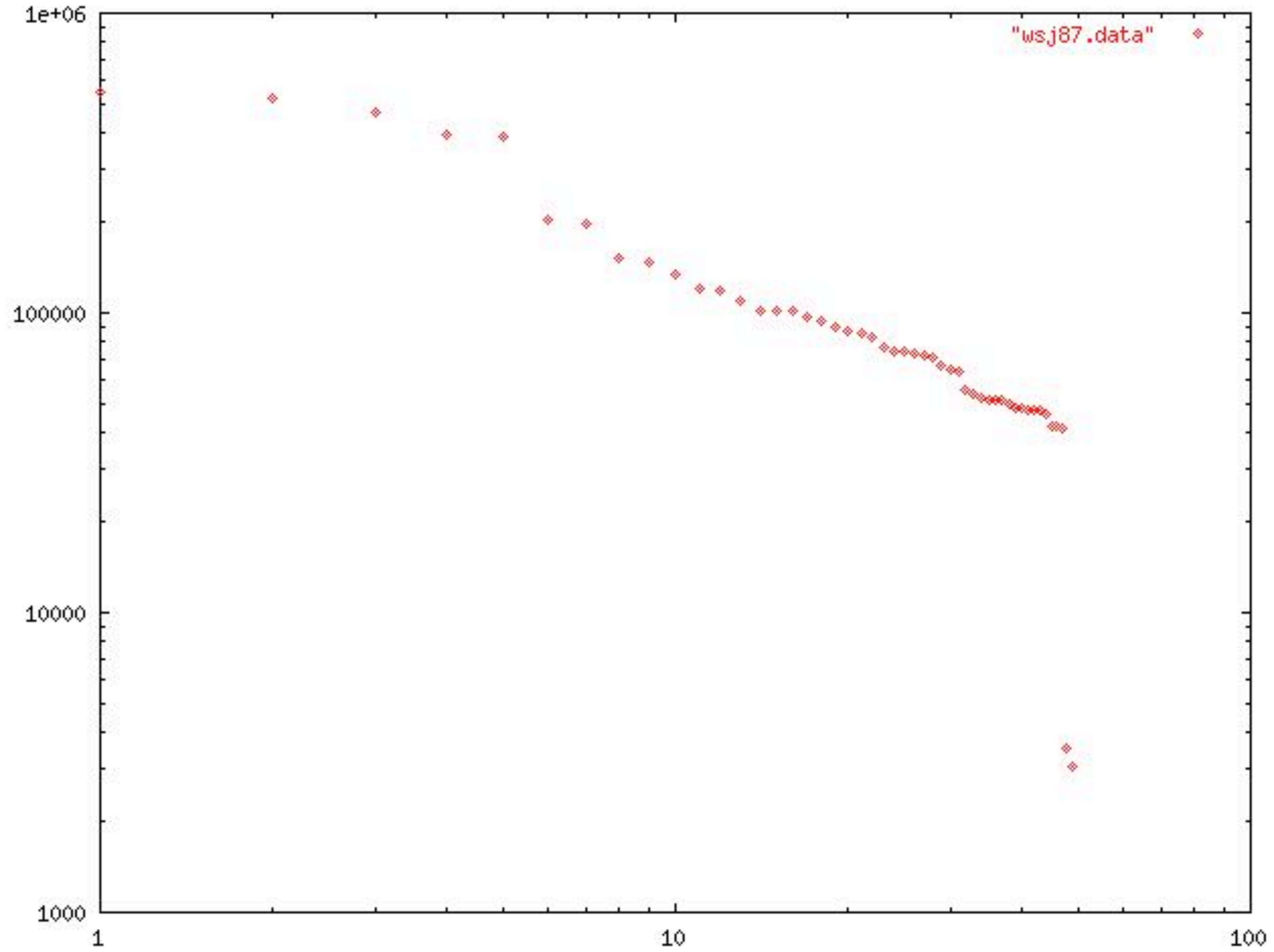
The collaboration graph is a power law graph, based on data from Math Review with 337451 authors with power 2.55

Collaboration graph (Math Review)

- 337,000 authors
- 496,000 edges
- Average 5.65 collaborations per person
- Average 2.94 collaborators per person
- Maximum degree 1401.   Guess who?
- A giant component of size 208,000
- 84,000 isolated vertices

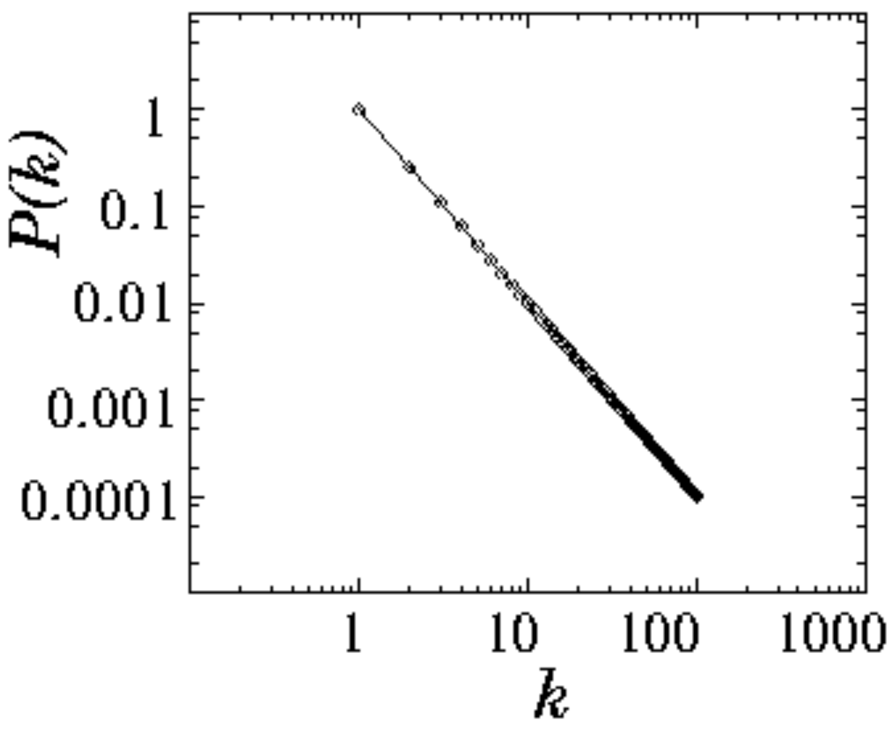


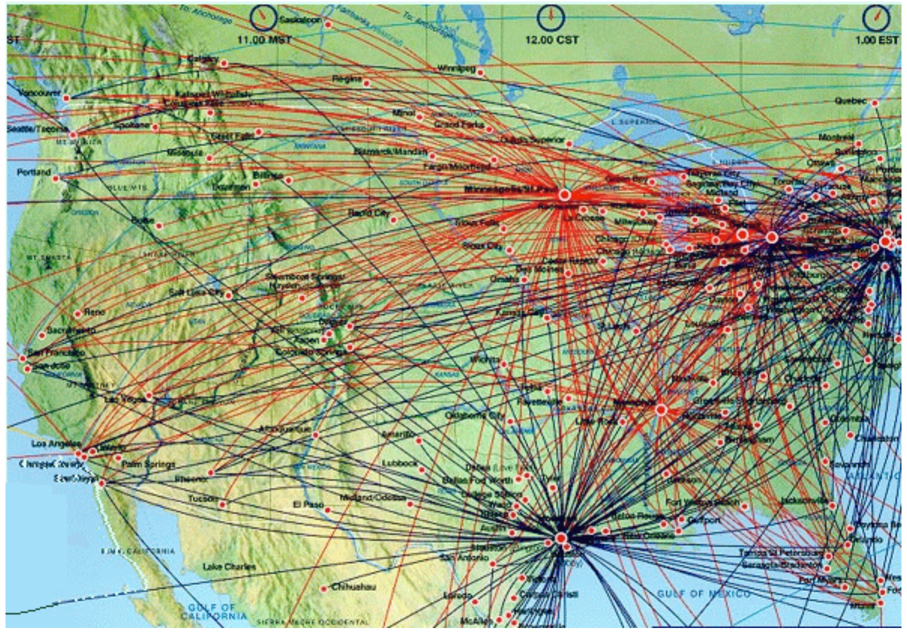
Ocurrences of words in TIME magazine articles
245412 terms.



Occurrences of words in WSJ Collection, a 131.6 MB collection of 46449 newspaper articles (19 million terms). Top 50 terms are included here.

Airline transportation networks are power graphs





Exponents for large power law networks

$$P(k) \sim k^{-\beta}$$

Networks	WWW	Actors	Citation Index	Power Grid	Phone calls
β	~2.1 (in) ~2.5 (out)	~2.3	~3	~4	~2.1

Numerous qustions

- What is a random graph? Which random graphs can best model real networks?
- Local growth rules versus global behavior?
- Communities and clustering
- network games, dynamics ……
- Applications-----   routing protocals
  biological networks
  network performance
  …

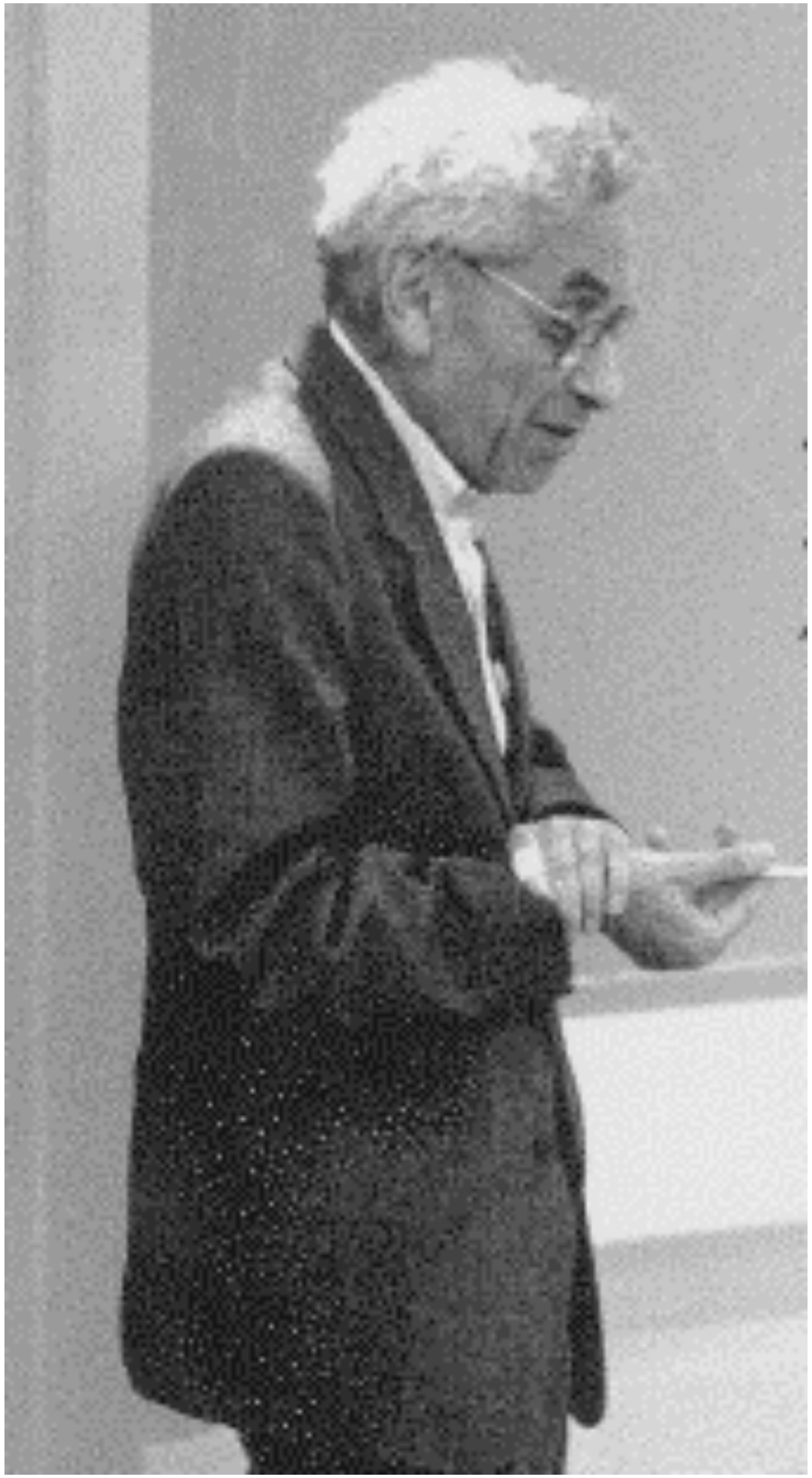
Questions:

- For a given sequence of integers, does it represent the degree sequence of some graph?

  Known. An old theorem of Erdos+Gallai 1960.

- For a given degree sequence of a subgraph, what is the mostly likely degree distribution of the host graph?

Hope I know! Depends on your random graph model!!



CSE 202

Matching algorithms

Fan Chung Graham

UC San Diego

An induced subgraph of the collaboration graph (with Erdos number at most 2).
Made by Fan Chung Graham and Lincoln Lu in 2002.

Gale-Sharpley Algorithm:

function stableMatching {

Initialize all *m* $\in$ M and *w* $\in$ W to *free*

while $\exists$ *free* man *m* who still has a woman w to propose to

{ w = m's highest ranked such woman

if w is *free,* (m, w) become *engaged*

else some pair (m', w) already exists

if w prefers m to m', (m, w) become *engaged* and

m' becomes *free*

else (m', w) remain *engaged*

}

}