Big Data Computing

Master's Degree in Computer Science

2021-2022

SAPIENZA
UNIVERSITÀ DI ROMA

Gabriele Tolomei

Department of Computer Science

Sapienza Università di Roma

tolomei@di.uniroma1.it

Analysis of (Large) Graphs

- Several "Big Data" tasks require us to work with large graphs

Analysis of (Large) Graphs

- Several "Big Data" tasks require us to work with large graphs
- Graph is a convenient abstraction to represent several data, e.g.:

Analysis of (Large) Graphs

- Several "Big Data" tasks require us to work with large graphs
- Graph is a convenient abstraction to represent several data, e.g.:
  - Web (i.e., the set of hyperlinked web pages)

Analysis of (Large) Graphs

- Several "Big Data" tasks require us to work with large graphs
- Graph is a convenient abstraction to represent several data, e.g.:
  - Web (i.e., the set of hyperlinked web pages)
  - Internet (i.e., the set of interconnected computers)

Analysis of (Large) Graphs

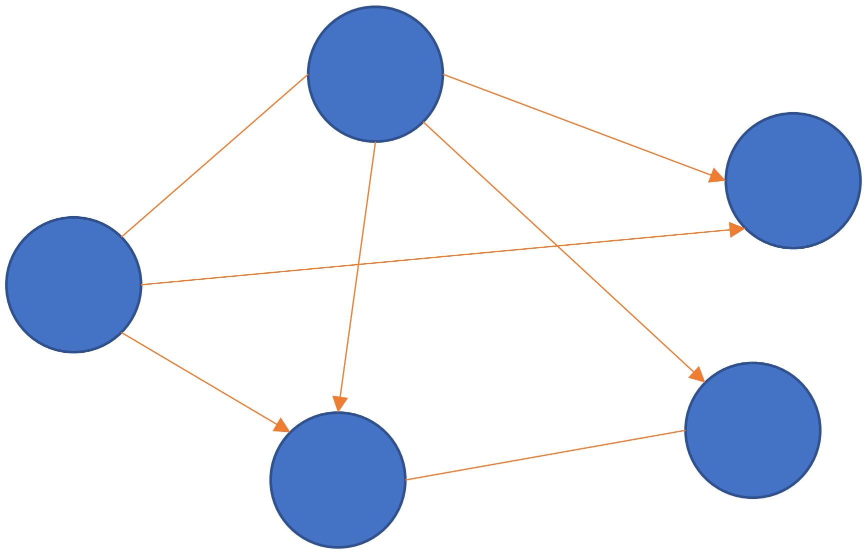
- Several "Big Data" tasks require us to work with large graphs
- Graph is a convenient abstraction to represent several data, e.g.:
  - Web (i.e., the set of hyperlinked web pages)
  - Internet (i.e., the set of interconnected computers)
  - Maps (i.e., the set of cities and roads connecting them)

Analysis of (Large) Graphs

- Several "Big Data" tasks require us to work with large graphs
- Graph is a convenient abstraction to represent several data, e.g.:
  - Web (i.e., the set of hyperlinked web pages)
  - Internet (i.e., the set of interconnected computers)
  - Maps (i.e., the set of cities and roads connecting them)
  - Social Networks (i.e., the set of social connections between people)
  - …

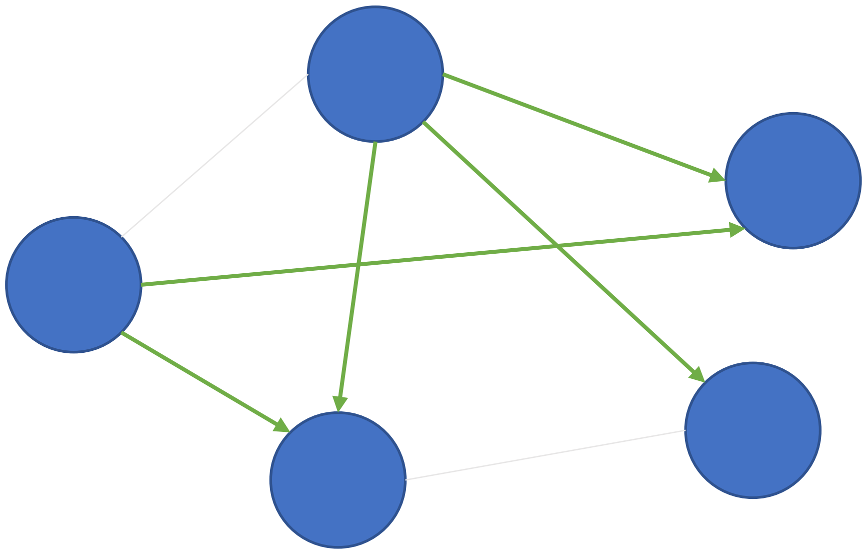
What is a Graph?

Informally, a set of vertices (nodes) connected by a set of edges (links)



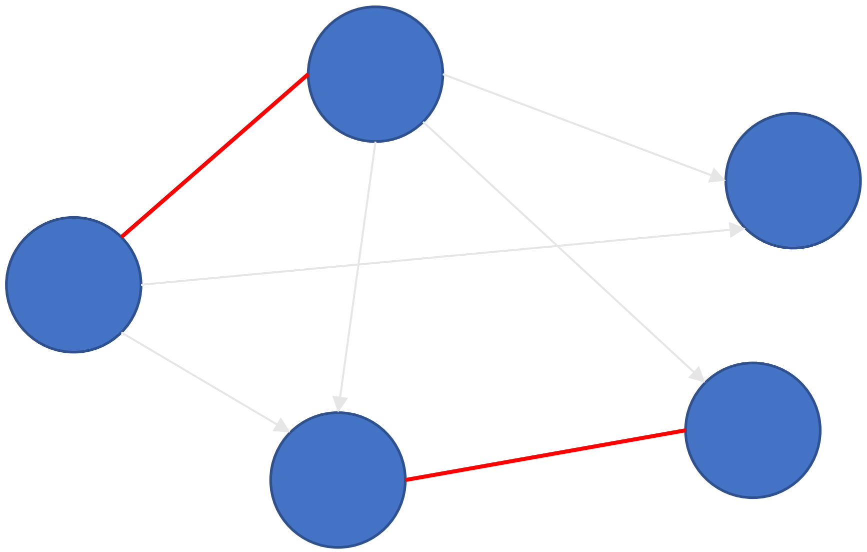
What is a Graph?

edges may be directed



What is a Graph?

edges may be undirected

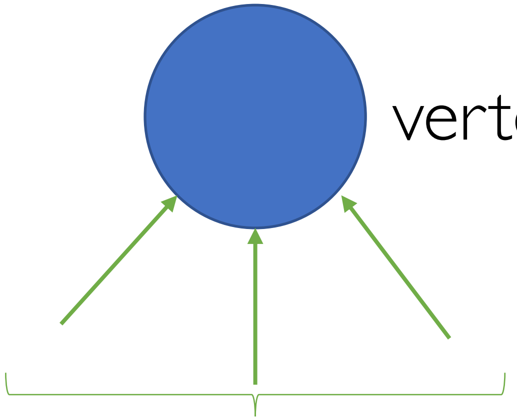


Directed vs. Undirected

Directed

Directed vs. Undirected

Directed

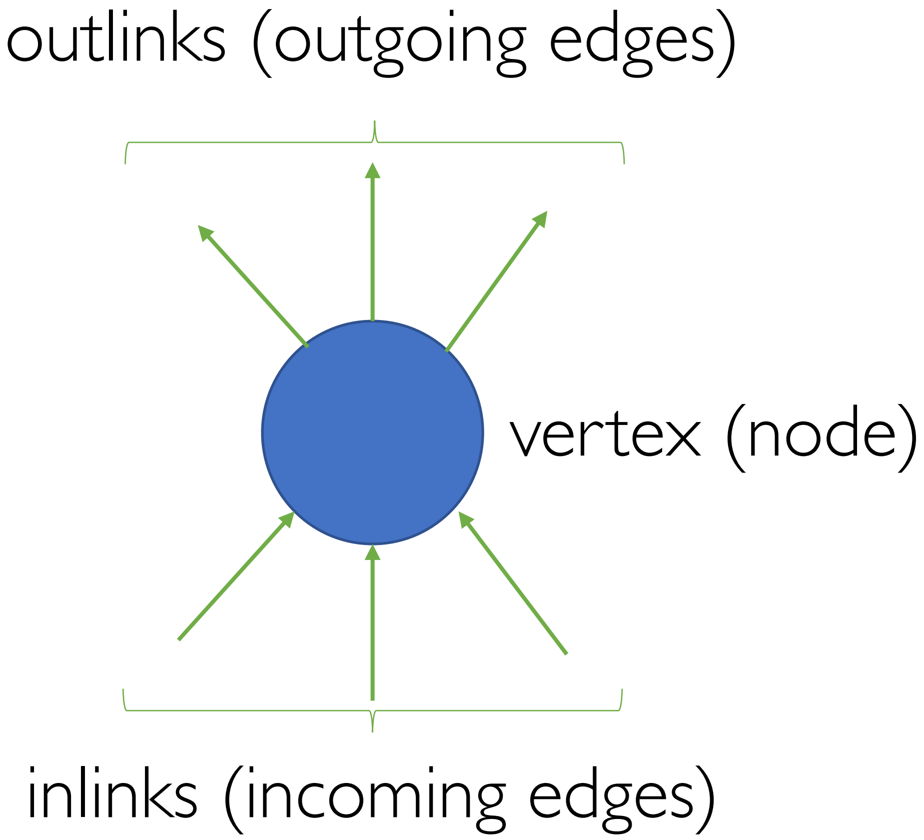


vertex (node)

inlinks (incoming edges)

Directed vs. Undirected

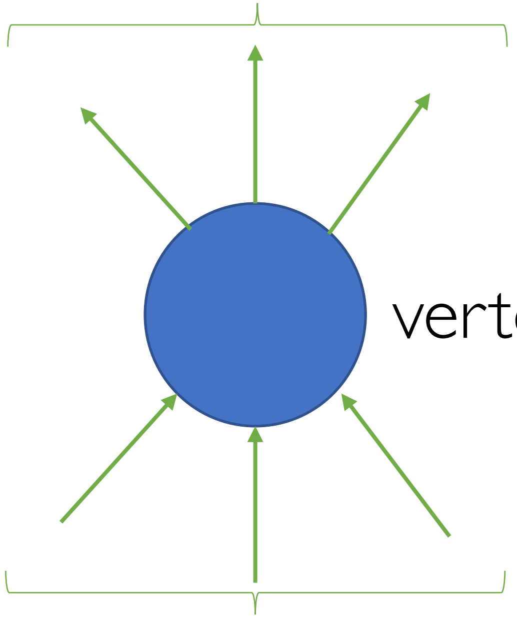
Directed



Directed vs. Undirected

Directed

outlinks (outgoing edges)



vertex (node)

inlinks (incoming edges)

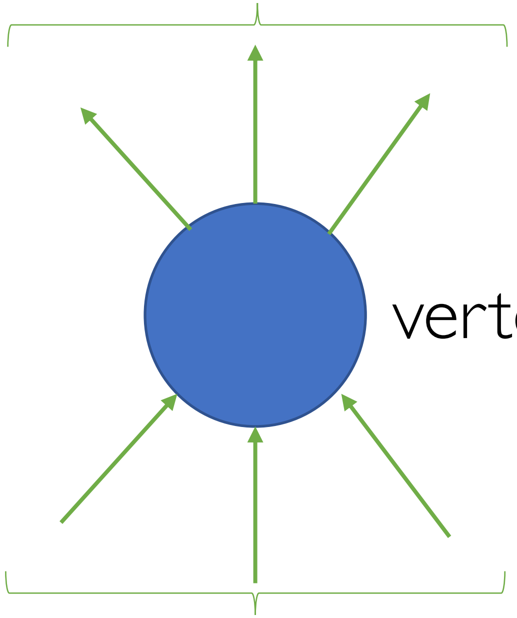
Undirected

vertex (node)

Directed vs. Undirected

Directed

outlinks (outgoing edges)

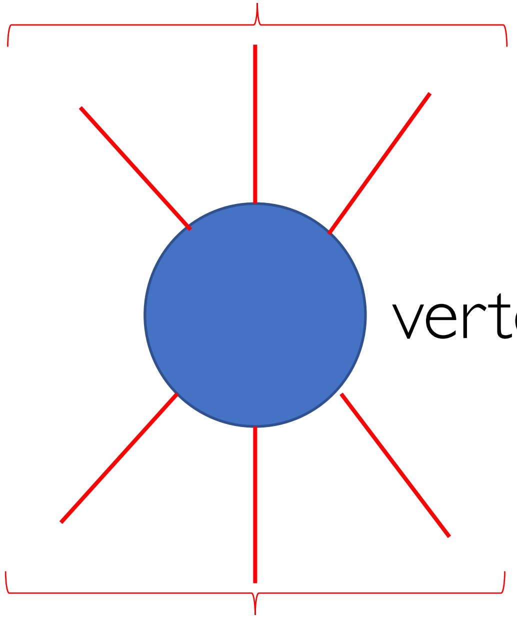


vertex (node)

inlinks (incoming edges)

Undirected

incident edges



vertex (node)

incident edges

Graph Formalism

$V = \{v_1, \ldots, v_n\}$ A set of nodes

Graph Formalism

$V = \{v_1, \ldots, v_n\}$ A set of nodes

$E \subseteq V \times V = \{(v_i, v_j) \in V \times V | v_i \neq v_j\}$ A set of edges

Graph Formalism

$V = \{v_1, \ldots, v_n\}$ A set of nodes

$E \subseteq V \times V = \{(v_i, v_j) \in V \times V | v_i \neq v_j\}$ A set of edges

$G = (V, E)$ A generic directed graph

Graph Formalism

$V = \{v_1, \ldots, v_n\}$ A set of nodes

$E \subseteq V \times V = \{(v_i, v_j) \in V \times V | v_i \neq v_j\}$ A set of edges

$G = (V, E)$ A generic directed graph

Note that an undirected graph is just a special case of a directed graph where the set of edges contain symmetric pairs of vertices

Node's Degree

Intuitively, the number of inbound/incident links to a node

Node's Degree

Intuitively, the number of inbound/incident links to a node

$$\deg(v) = |\{u \in V | (u, v) \in E\}|$$

Node's Degree

Intuitively, the number of inbound/incident links to a node

$$\deg(v) = |\{u \in V | (u, v) \in E\}|$$

To be more explicit, in the case of a directed graph sometimes we distinguish between in-degree and out-degree

$$\text{in-deg}(v) = |\{u \in V | (u, v) \in E\}|$$

$$\text{out-deg}(v) = |\{u \in V | (v, u) \in E\}|$$

How Do We Represent Graphs?

3 main ways of representing graphs

How Do We Represent Graphs?

3 main ways of representing graphs

Adjacency Matrices

How Do We Represent Graphs?

3 main ways of representing graphs

Adjacency Matrices

Adjacency Lists

How Do We Represent Graphs?

3 main ways of representing graphs

Adjacency Matrices

Adjacency Lists

Edge Lists

Adjacency Matrix

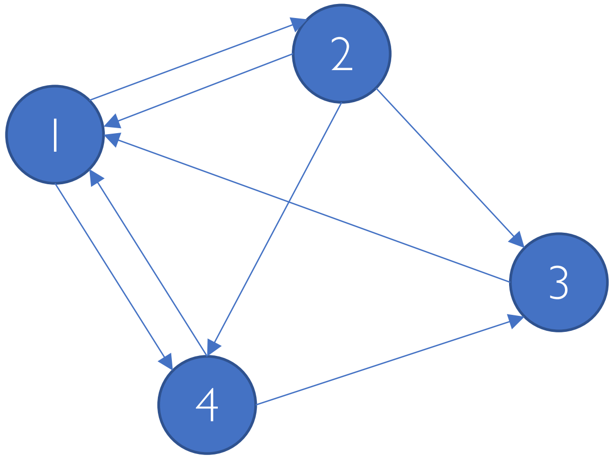
- Given a graph G = (V, E) with |V| = n vertices

Adjacency Matrix

- Given a graph G = (V, E) with |V| = n vertices
- Build an n-by-n square matrix M where:
  - M[i, j] = 1 iff there exists an edge from vertex v_i to vertex v_j

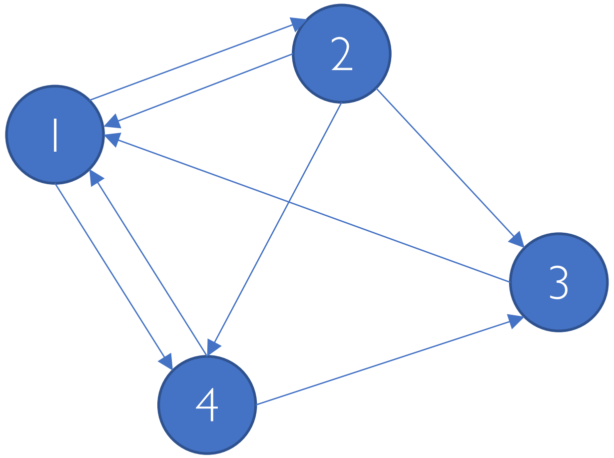
Adjacency Matrix

- Given a graph G = (V, E) with |V| = *n* vertices
- Build an *n*-by-*n* square matrix M where:
  - M[i, j] = 1 iff there exists an edge from vertex v_i to vertex v_j



Adjacency Matrix

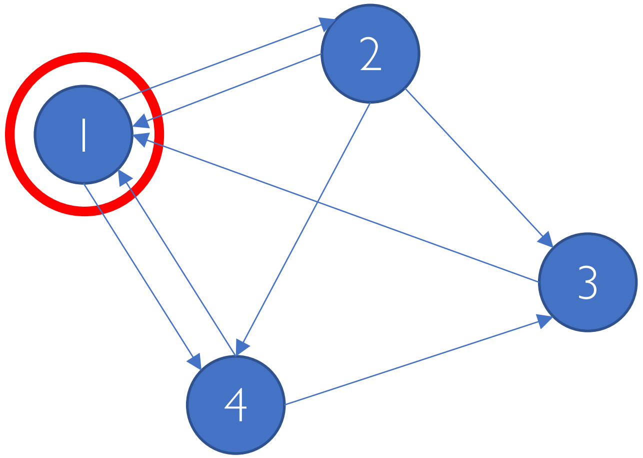
- Given a graph G = (V, E) with |V| = n vertices
- Build an n-by-n square matrix M where:
  - M[i, j] = 1 iff there exists an edge from vertex v_i to vertex v_j



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

Adjacency Matrix

- Given a graph G = (V, E) with |V| = n vertices
- Build an n-by-n square matrix M where:
  - M[i, j] = 1 iff there exists an edge from vertex v_i to vertex v_j



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

Adjacency Matrix: PROs and CONs

- PROs:
  - Most intuitive representation
  - Ready-to-go for mathematical manipulation

Adjacency Matrix: PROs and CONs

- PROs:
  - Most intuitive representation
  - Ready-to-go for mathematical manipulation
- CONs:
  - Space inefficient (especially for loosely connected graphs, i.e., sparse matrices)
  - Easy to write yet hard to compute

Adjacency Lists

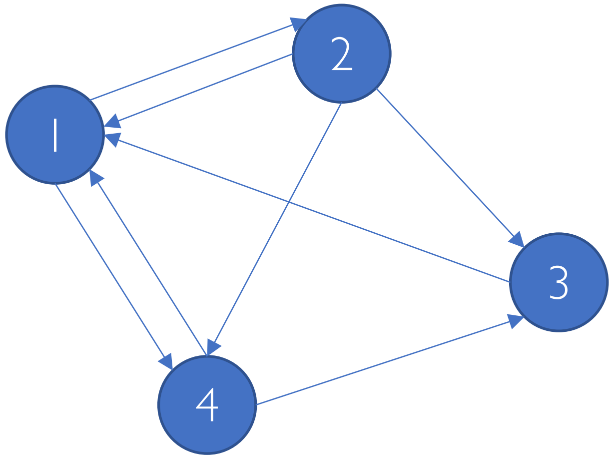
- Take the adjacency matrix and throw away all the 0s

Adjacency Lists

- Take the adjacency matrix and throw away all the 0s
- Associate with each node a linked list whose head is the node and each pointer points to an adjacent vertex of the head's node

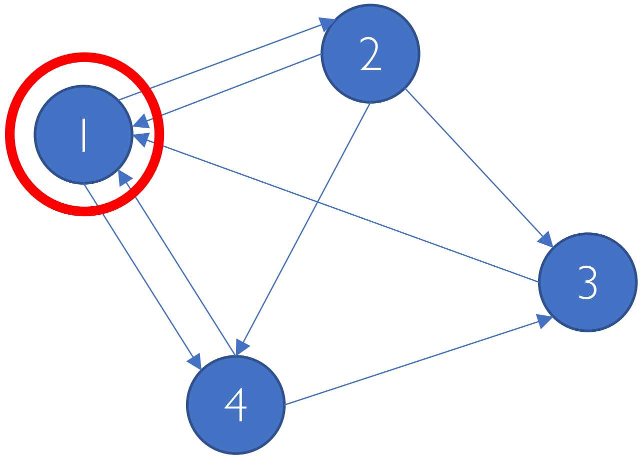
Adjacency Lists

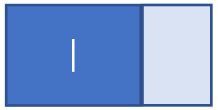
- Take the adjacency matrix and throw away all the 0s
- Associate with each node a linked list whose head is the node and each pointer points to an adjacent vertex of the head's node



Adjacency Lists

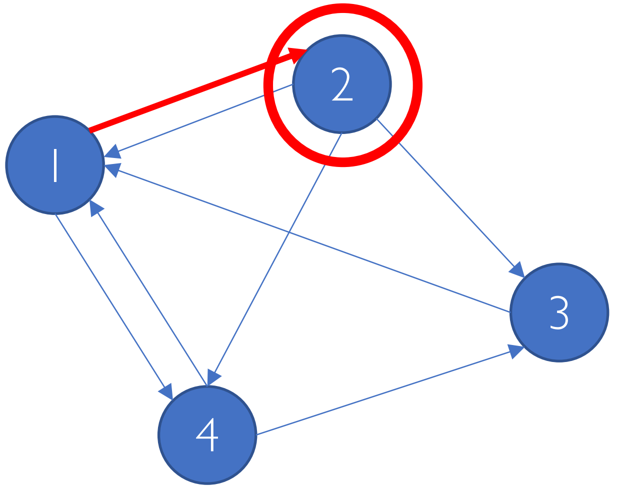
- Take the adjacency matrix and throw away all the 0s
- Associate with each node a linked list whose head is the node and each pointer points to an adjacent vertex of the head's node

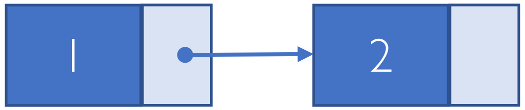




Adjacency Lists

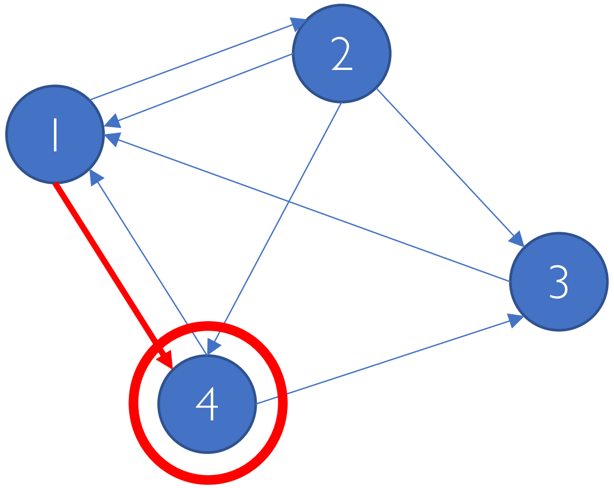
- Take the adjacency matrix and throw away all the 0s
- Associate with each node a linked list whose head is the node and each pointer points to an adjacent vertex of the head's node

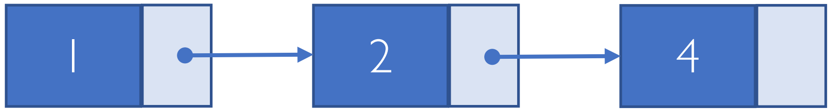




Adjacency Lists

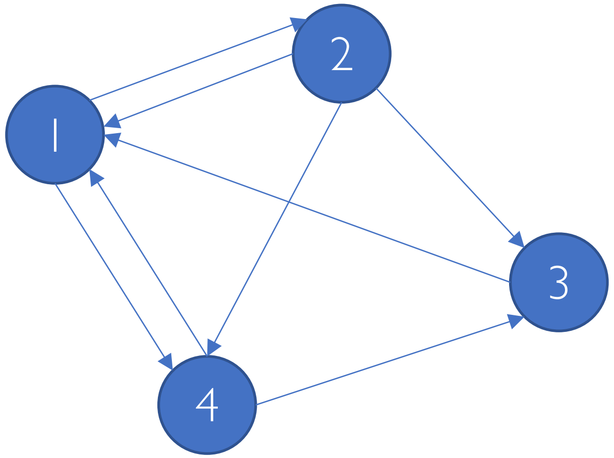
- Take the adjacency matrix and throw away all the 0s
- Associate with each node a linked list whose head is the node and each pointer points to an adjacent vertex of the head's node

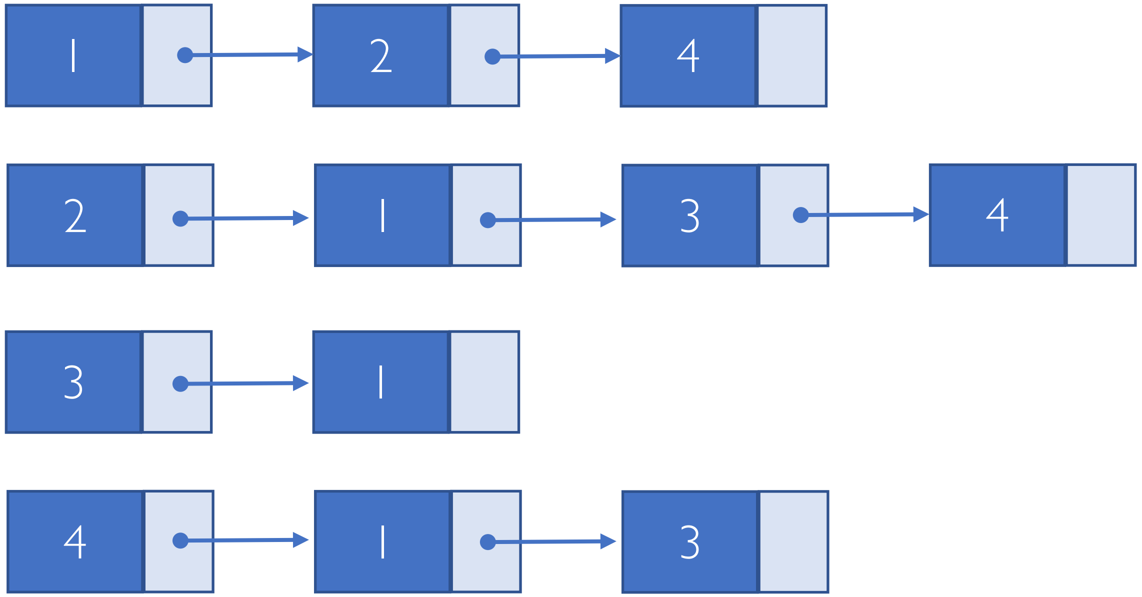




Adjacency Lists

- Take the adjacency matrix and throw away all the 0s
- Associate with each node a linked list whose head is the node and each pointer points to an adjacent vertex of the head's node





Adjacency Lists: PROs and CONs

- PROs:
  - Compact representation (compression)
  - Easy to compute anything over outgoing links

Adjacency Lists: PROs and CONs

- PROs:
  - Compact representation (compression)
  - Easy to compute anything over outgoing links
- CONs:
  - Hard to compute anything over incoming links

Adjacency Lists: PROs and CONs

- PROs:
  - Compact representation (compression)
  - Easy to compute anything over outgoing links
- CONs:
  - Hard to compute anything over incoming links

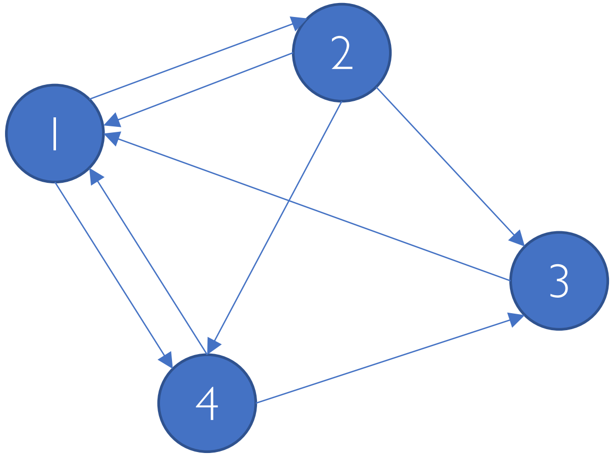
Note that with adjacency matrix, any computation over incoming (outgoing) links reduces to a column (row) scan of the matrix

Edge Lists

- Explicitly enumerates all the edges

Edge Lists

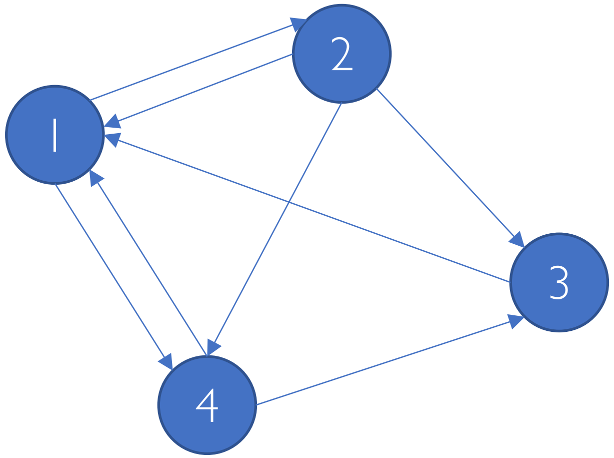
- Explicitly enumerates all the edges



(1, 2)	(2, 1)	(3, 1)	(4, 1)
(1, 4)	(2, 3)		(4, 3)
	(2, 4)		

Edge Lists

- Explicitly enumerates all the edges



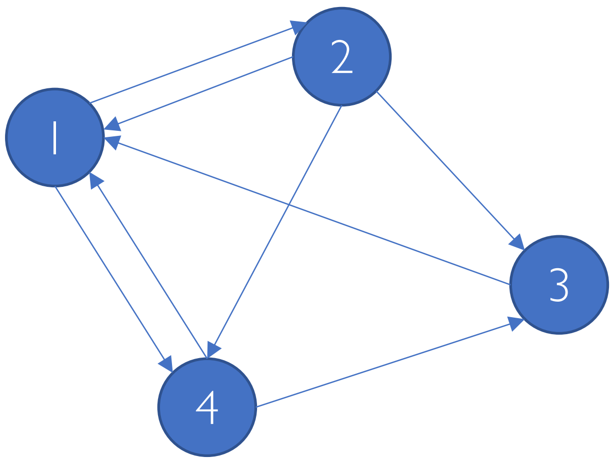
(1, 2) (2, 1) (3, 1) (4, 1)
(1, 4) (2, 3) (4, 3)
(2, 4)

PROs

Easily support for edge insertions

Edge Lists

- Explicitly enumerates all the edges



(1, 2)	(2, 1)	(3, 1)	(4, 1)
(1, 4)	(2, 3)		(4, 3)
	(2, 4)		

PROs

Easily support for edge insertions

CONs

Waste of space

Some Famous Graph Problems

Problems

Applications

Some Famous Graph Problems

Problems	Applications
Finding Shortest Paths	Routing IP packets, GPS navigation systems

Some Famous Graph Problems

Problems	Applications
Finding Minimum Spanning Tree	Telco laying down fiber cables

Some Famous Graph Problems

Problems

Finding Max Flow

Applications

Airline scheduling

Some Famous Graph Problems

Problems	Applications
Identifying special nodes or subgraphs	Community detection in social networks

Some Famous Graph Problems

Problems	Applications
Link Analysis	Web page ranking

Some Famous Graph Problems

Problems	Applications
Finding Shortest Paths	Routing IP packets, GPS navigation systems
Finding Minimum Spanning Tree	Telco laying down fiber cables
Finding Max Flow	Airline scheduling
Identifying special nodes or subgraphs	Community detection in social networks
Link Analysis	Web page ranking

(Graph) Link Analysis

- A data analysis technique used to evaluate relationships (connections) between nodes of a graph

(Graph) Link Analysis

- A data analysis technique used to evaluate relationships (connections) between nodes of a graph
- The idea is to extrapolate useful patterns/information out of the structure of the graph only

(Graph) Link Analysis

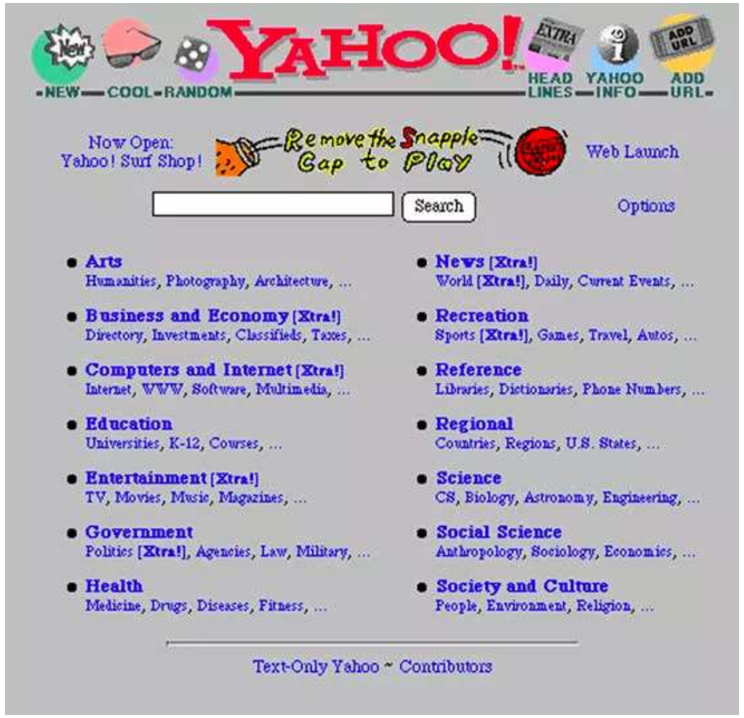
- A data analysis technique used to evaluate relationships (connections) between nodes of a graph
- The idea is to extrapolate useful patterns/information out of the structure of the graph only
- The Web graph is a great test bed for link analysis

Web Directories

- Around mid 90's people started to think about how web pages on the Internet should be organized

Web Directories

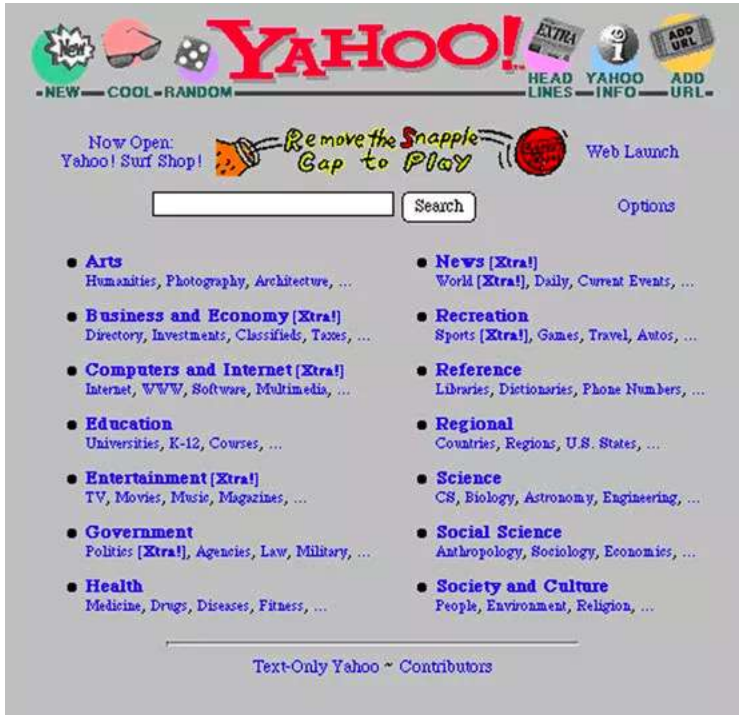
- Around mid 90's people started to think about how web pages on the Internet should be organized
- A first attempt made by Yahoo! in 1994 was to organize web pages into a set of human-curated categories



Yet Another Hierarchical Officious Oracle

Web Directories

- Around mid 90's people started to think about how web pages on the Internet should be organized
- A first attempt made by Yahoo! in 1994 was to organize web pages into a set of human-curated categories
- Other attempts: DMOZ, LookSmart



Yet Another Hierarchical Officious Oracle

Web Information Retrieval

- Very soon, this manually-curated categorization doesn't scale to the size of the fast-growing Web

Web Information Retrieval

- Very soon, this manually-curated categorization **doesn't scale** to the size of the fast-growing Web
- Forget about fixed web directories!

Web Information Retrieval

- Very soon, this manually-curated categorization **doesn't scale** to the size of the fast-growing Web
- Forget about fixed web directories!
- The Web can be seen as a **huge corpus** of documents (i.e., web pages)

Web Information Retrieval

- Very soon, this manually-curated categorization doesn't scale to the size of the fast-growing Web
- Forget about fixed web directories!
- The Web can be seen as a huge corpus of documents (i.e., web pages)
- Let the users search for relevant web documents using natural language queries through information retrieval techniques

Web Information Retrieval

- Very soon, this manually-curated categorization doesn't scale to the size of the fast-growing Web
- Forget about fixed web directories!
- The Web can be seen as a huge corpus of documents (i.e., web pages)
- Let the users search for relevant web documents using natural language queries through information retrieval techniques

Web Search Engines

Web Information Retrieval

- Following traditional IR approach, the first web search engines were designed to find relevant web documents using content only

Web Information Retrieval

- Following traditional IR approach, the first web search engines were designed to find relevant web documents using content only
- Both queries and documents were mapped to the same word space

Web Information Retrieval

- Following traditional IR approach, the first web search engines were designed to find relevant web documents using content only
- Both queries and documents were mapped to the same word space
- Each word of a document is scored on the basis of its importance within that document and overall the corpus (e.g., TF-IDF)

Web Information Retrieval

- Following traditional IR approach, the first web search engines were designed to find relevant web documents using content only
- Both queries and documents were mapped to the same word space
- Each word of a document is scored on the basis of its importance within that document and overall the corpus (e.g., TF-IDF)
- The list of top-*k* documents most similar to a query are returned (e.g., measuring cosine similarity between each query-document pair)

Web Information Retrieval

Traditional IR applied to web documents suffers from 2 main problems

Web Information Retrieval

Traditional IR applied to web documents suffers from 2 main problems

information overload

For the query "`Barack Obama`" Google retrieves more than 150M relevant web pages
Result pages contain only the top-10 most relevant ones
How could we rank them all?

Web Information Retrieval

Traditional IR applied to web documents suffers from 2 main problems

result trustworthiness

Traditional IR is designed to work on small, trusted collections of documents (e.g., curated digital libraries) Content-based similarity can be hacked by artificially creating web documents which are just spam

Web Information Retrieval

Traditional IR applied to web documents suffers from 2 main problems

information overload

For the query "`Barack Obama`" Google retrieves more than 150M relevant web pages
Result pages contain only the top-10 most relevant ones
How could we rank them all?

result trustworthiness

Traditional IR is designed to work on small, trusted collections of documents (e.g., curated digital libraries)
Content-based similarity can be hacked by artificially creating web documents which are just spam

The Web is huge and full of untrusted documents!

Web Search Challenges

We need a way to assess the trustworthiness/importance
of a web page from the structure of the Web graph

Web Search Challenges

We need a way to assess the trustworthiness/importance of a web page from the structure of the Web graph

Web pages that are pointed to by many other pages are likely to contain authoritative information

Web Search Challenges

We need a way to assess the trustworthiness/importance of a web page from the structure of the Web graph

Web pages that are pointed to by many other pages are likely to contain authoritative information

Trustworthy web pages should point to each other

Ranking Nodes of the Web Graph

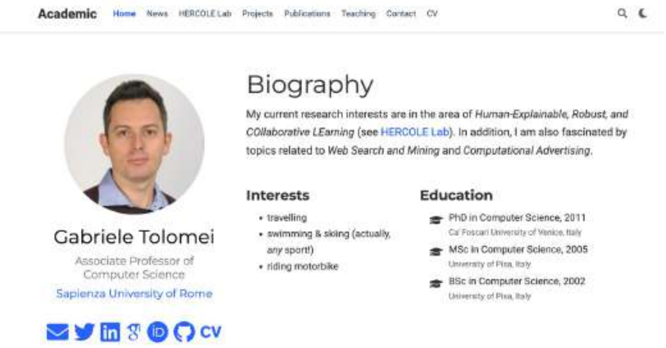
All web pages are not created equal!

Ranking Nodes of the Web Graph

All web pages are not created equal!

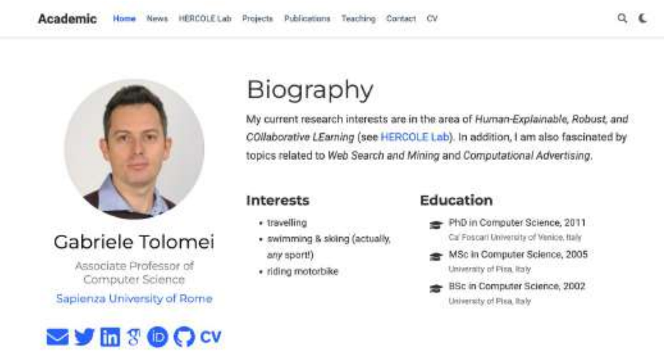
Ranking Nodes of the Web Graph

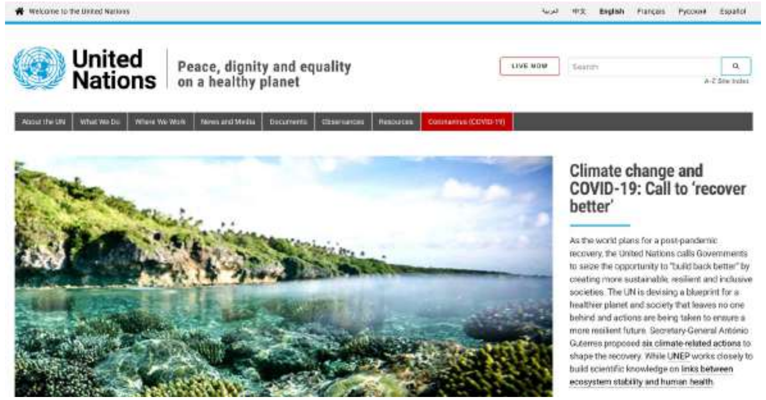
All web pages are not created equal!



Ranking Nodes of the Web Graph

All web pages are not created equal!





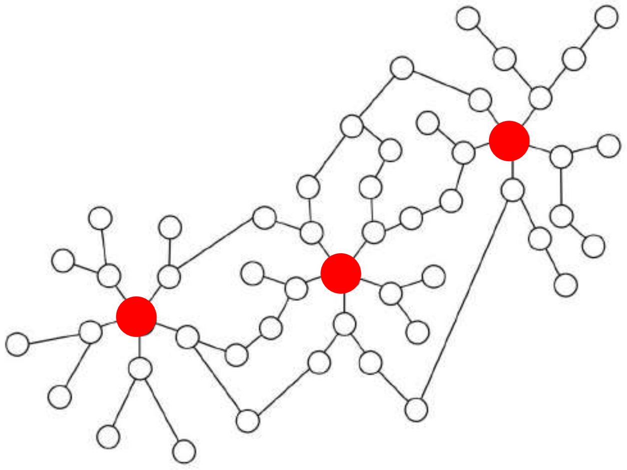
Ranking Nodes of the Web Graph

If we look at the Web graph we will see a huge difference between each node's connectivity

Ranking Nodes of the Web Graph

If we look at the Web graph we will see a huge difference between each node's connectivity

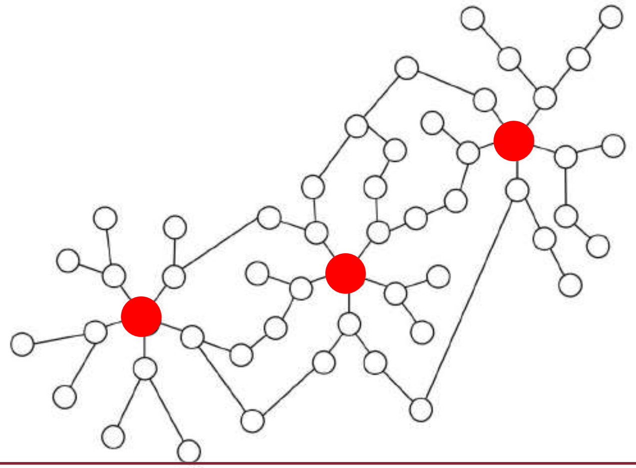
Some nodes appear to be more "connected" than others



Ranking Nodes of the Web Graph

If we look at the Web graph we will see a huge **difference** between each **node's connectivity**

Some nodes appear to be more "connected" than others



Rank nodes (i.e., assign them an importance score) on the basis of their connectivity

The Web as a "Scale-Free" Network

In 1999, Barabasi-Albert used a web crawler to map the connectedness of a portion of the Web

The Web as a "Scale-Free" Network

In 1999, Barabasi-Albert used a web crawler to map the connectedness of a portion of the Web

They compared the degree distribution of a randomly generated graph with that observed from the crawled Web

The Web as a "Scale-Free" Network

In 1999, Barabasi-Albert used a web crawler to map the connectedness of a portion of the Web

They compared the degree distribution of a randomly generated graph with that observed from the crawled Web

They observed the degree distribution follows a power law

The Web as a "Scale-Free" Network

In 1999, Barabasi-Albert used a web crawler to map the connectedness of a portion of the Web

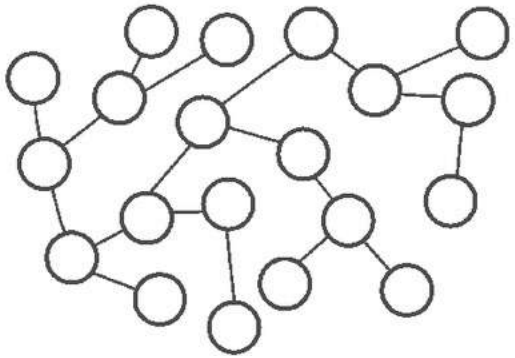
They compared the degree distribution of a randomly generated graph with that observed from the crawled Web

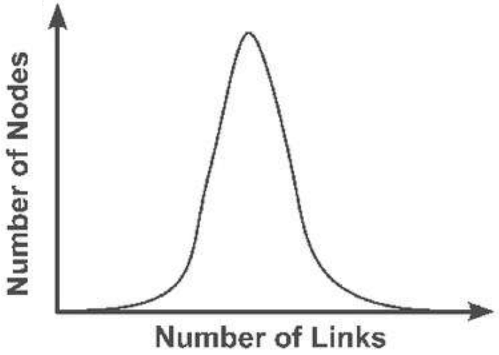
They observed the degree distribution follows a power law

They refer to graphs (i.e., networks) exhibiting such a behavior as scale-free networks

Barabási, A.L. and Albert,R. *Emergence of scaling in random networks*, Science, 286:509-512, October 15, 1999

The Web as a Scale-Free Network

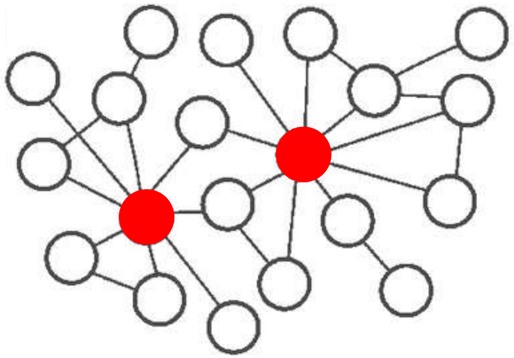


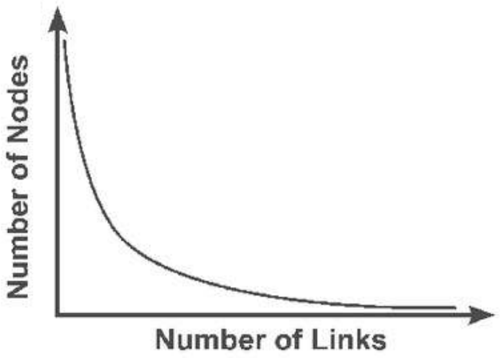


Random Graph

Most nodes have approximately the same number of links producing a bell-shaped curve of the degree distribution

The Web as a Scale-Free Network

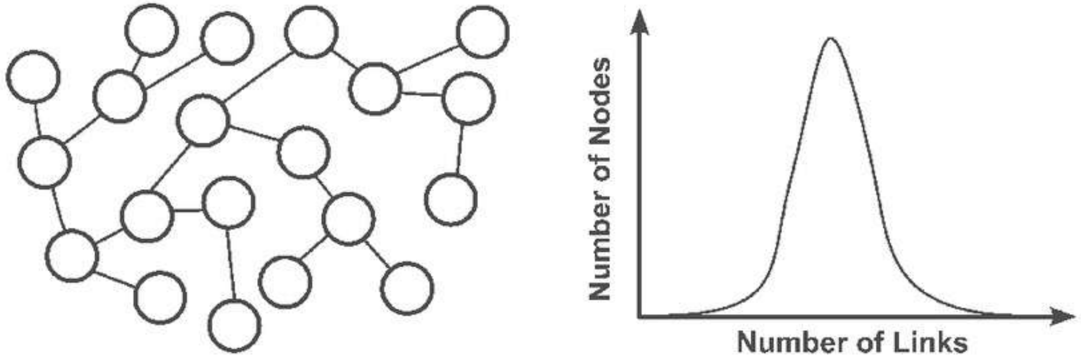




Scale-Free Graph

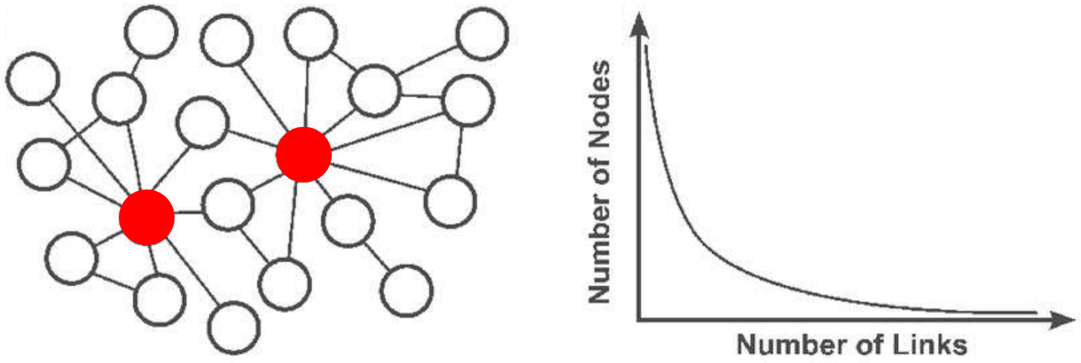
Most nodes have few links, and few nodes (i.e., red ones) have a large number of links, resulting into a power law degree distribution

The Web as a Scale-Free Network



Random Graph

Most nodes have approximately the same number of links producing a bell-shaped curve of the degree distribution



Scale-Free Graph

Most nodes have few links, and few nodes (i.e., red ones) have a large number of links, resulting into a power law degree distribution

Tëmkin, I. and Eldredge N. *Networks and Hierarchies: Approaching Complexity in Evolutionary Theory*. February 2015

Scale-Free Networks

The fraction of nodes in the network having *k* connections to other nodes follow a power law distribution

$$\boxed{P(\deg = k) = p(k) = \alpha k^{-\gamma} \propto k^{-\gamma}}$$

Scale-Free Networks

The fraction of nodes in the network having *k* connections to other nodes follow a power law distribution

$$\boxed{P(\deg = k) = p(k) = \alpha k^{-\gamma} \propto k^{-\gamma}}$$

80÷20 Pareto principle

Roughly 80% of the effects come from 20% of the causes

Scale-Free Networks

The fraction of nodes in the network having *k* connections to other nodes follow a power law distribution

$$\boxed{P(\deg = k) = p(k) = \alpha k^{-\gamma} \propto k^{-\gamma}}$$

80÷20 Pareto principle

Roughly 80% of the effects come from 20% of the causes

The ratio of very connected nodes to the number of nodes in the rest of the network remains constant as the network changes in size

Why "Scale-Free"?

A power law looks the same, no matter what scale we look at it

Why "Scale-Free"?

A power law looks the same, no matter what scale we look at it

The shape of the distribution is unchanged, except for a multiplicative constant

Why "Scale-Free"?

A power law looks the same, no matter what scale we look at it

The shape of the distribution is unchanged, except for a multiplicative constant

A generic probability distribution $p(x)$ is scale-free if it exists $g(c)$
s.t. $p(cx) = g(c)p(x)$ for each c and x

Why "Scale-Free"?

A power law looks the same, no matter what scale we look at it

The shape of the distribution is unchanged, except for a multiplicative constant

A generic probability distribution $p(x)$ is scale-free if it exists $g(c)$
s.t. $p(cx) = g(c)p(x)$ for each c and x

For the power law: $$p(x) = \alpha x^{-\gamma}$$

Why "Scale-Free"?

A power law looks the same, no matter what scale we look at it

The shape of the distribution is unchanged, except for a multiplicative constant

A generic probability distribution *p(x)* is scale-free if it exists *g(c)* s.t. *p(cx)* = *g(c)p(x)* for each c and *x*

For the power law: $$p(x) = \alpha x^{-\gamma}$$

$$p(cx) = \alpha(cx)^{-\gamma} = c^{-\gamma}\alpha x^{-\gamma}$$

Why "Scale-Free"?

A power law looks the same, no matter what scale we look at it

The shape of the distribution is unchanged, except for a multiplicative constant

A generic probability distribution *p(x)* is scale-free if it exists *g(c)*
s.t. *p(cx)* = *g(c)p(x)* for each c and *x*

For the power law: $p(x) = \alpha x^{-\gamma}$

$$p(cx) = \alpha(cx)^{-\gamma} = c^{-\gamma}\alpha x^{-\gamma} = \boxed{g(c)p(x) \quad g(c) = c^{-\gamma}}$$

Preferential Attachment

A scale-free network can be constructed by progressively adding nodes to an existing network

Preferential Attachment

A scale-free network can be constructed by progressively adding nodes to an existing network

Links to existing nodes are created following the preferential attachment (i.e., rich get richer) principle

Preferential Attachment

A scale-free network can be constructed by progressively adding nodes to an existing network

Links to existing nodes are created following the preferential attachment (i.e., rich get richer) principle

The probability that the new node is linked to an existing node i is proportional to the number of existing links k_i that node i already has

Preferential Attachment

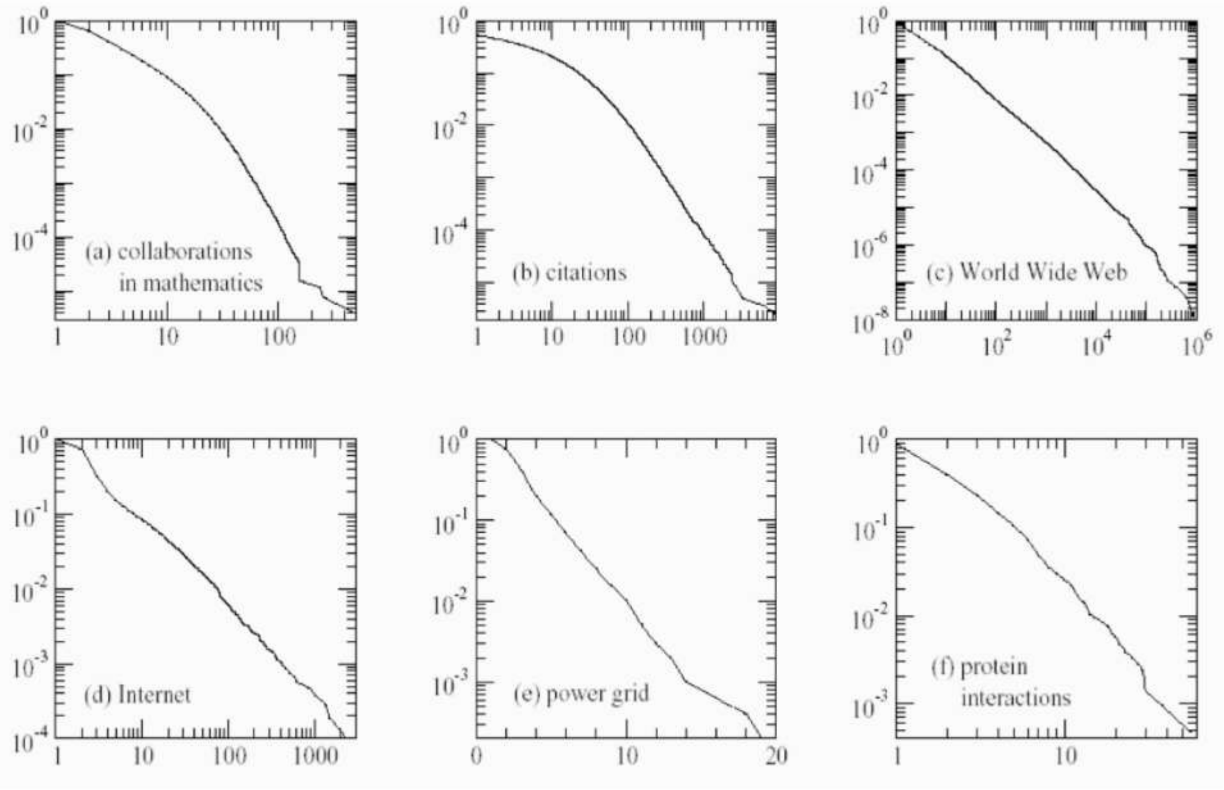
A scale-free network can be constructed by progressively adding nodes to an existing network

Links to existing nodes are created following the preferential attachment (i.e., rich get richer) principle

The probability that the new node is linked to an existing node *i* is proportional to the number of existing links k_i that node *i* already has

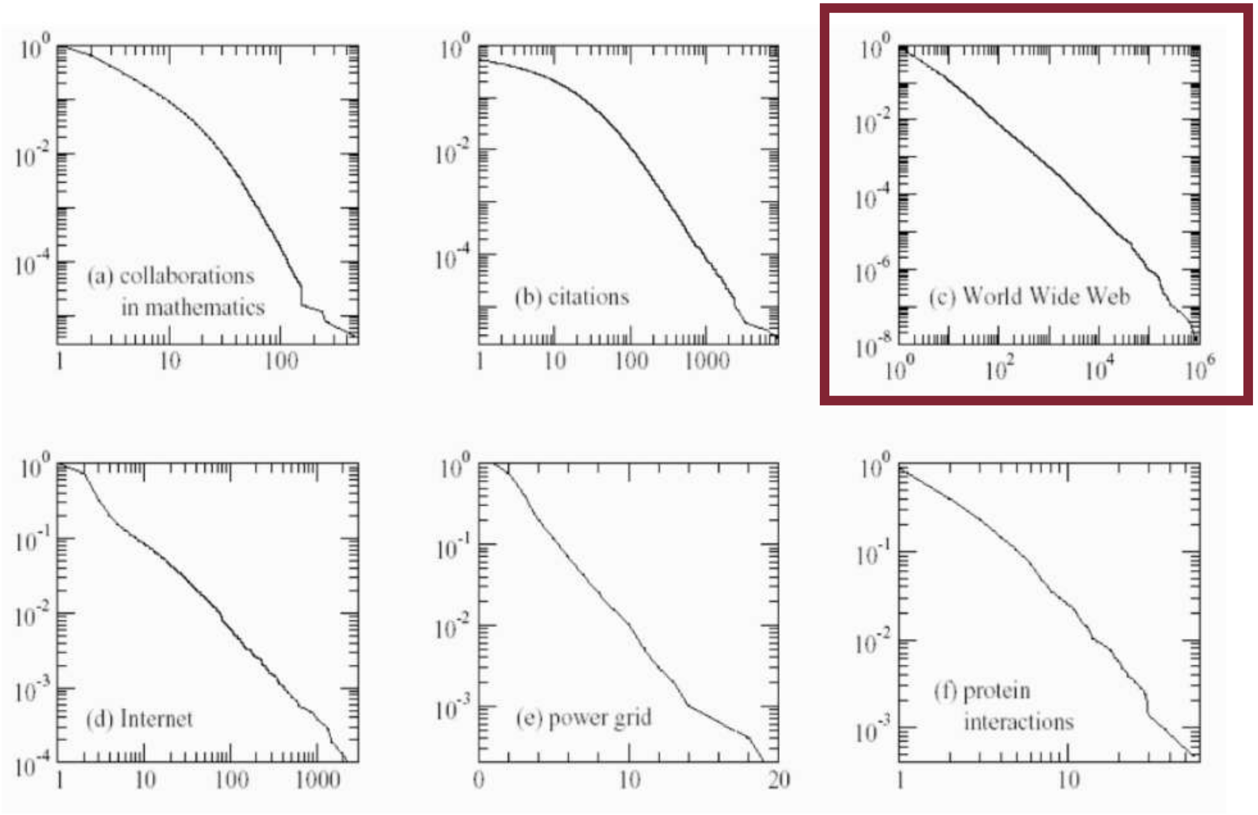
$$\boxed{p(\text{linking to node } i) \propto \frac{k_i}{\sum_j k_j}}$$

Scale-Free Networks: Examples



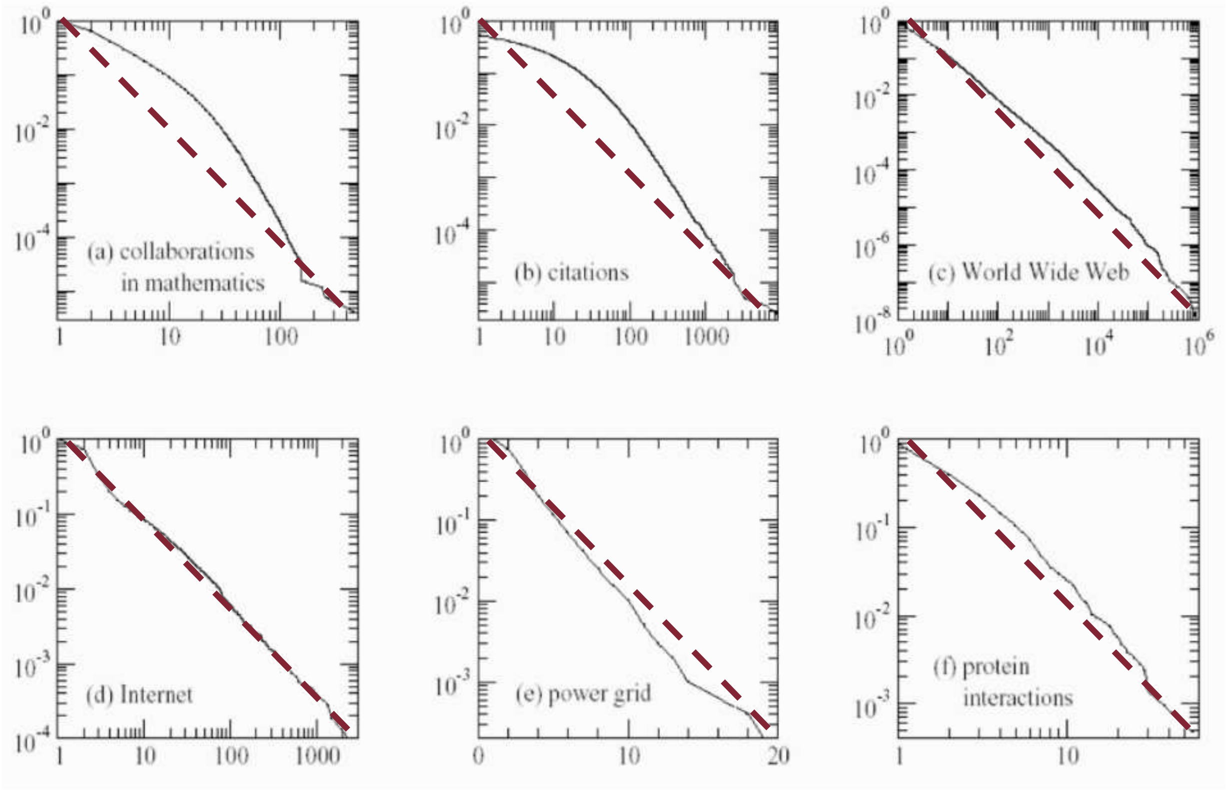
Many real-world networks are scale-free

Scale-Free Networks: Examples



The Web is one of those!

Scale-Free Networks: Examples



On log-log scale power law distributions look like straight lines

$$\log(p(k)) = \log(\alpha k^{-\gamma}) = \underbrace{\log(\alpha)}_{\text{constant } q} + \log(k^{-\gamma}) = q - \gamma \log(k)$$

Computing Node Importance

Several link analysis approaches to compute web page importance

Computing Node Importance

Several link analysis approaches to compute web page importance

PageRank

Computing Node Importance

Several link analysis approaches to compute web page importance

PageRank

Hubs and Authorities (HITS)

Computing Node Importance

Several link analysis approaches to compute web page importance

PageRank

Hubs and Authorities (HITS)

Personalized PageRank

Computing Node Importance

Several link analysis approaches to compute web page importance

PageRank

Hubs and Authorities (HITS)

Personalized PageRank

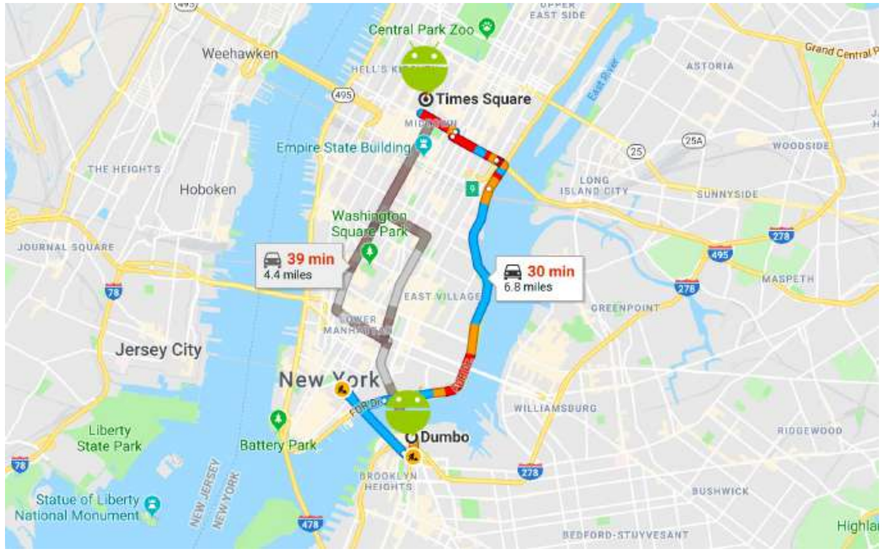
Web Spam Detection

Take-Home Message of Today

- Many Big Data tasks require to work with data which naturally resembles the structure of a graph

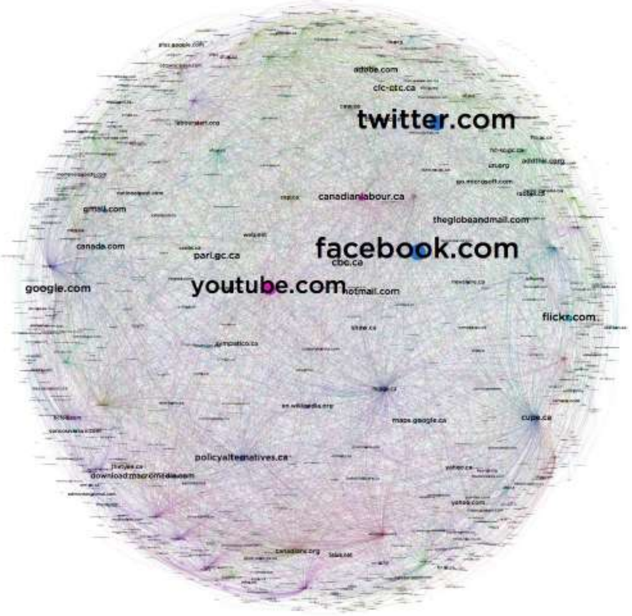
Take-Home Message of Today

- Many Big Data tasks require to work with data which naturally resembles the structure of a graph
- Examples:
    - Finding the shortest path on a map



Take-Home Message of Today

- Many Big Data tasks require to work with data which naturally resembles the structure of a graph
- Examples:
  - Finding the shortest path on a map
  - Computing the importance of a page in the Web graph



Take-Home Message of Today

- Many Big Data tasks require to work with data which naturally resembles the structure of a graph
- Examples:
  - Finding the shortest path on a map
  - Computing the importance of a page in the Web graph
  - Suggesting friends in a social network graph

Take-Home Message of Today

- Many Big Data tasks require to work with data which naturally resembles the structure of a graph
- Examples:
  - Finding the shortest path on a map
  - Computing the importance of a page in the Web graph
  - Suggesting friends in a social network graph
- Several algorithms and techniques exist to approach the problems above

Take-Home Message of Today

- Many Big Data tasks require to work with data which naturally resembles the structure of a graph
- Examples:
  - Finding the shortest path on a map
  - Computing the importance of a page in the Web graph
  - Suggesting friends in a social network graph
- Several algorithms and techniques exist to approach the problems above
- Working with large-scale graphs may require specific tools/frameworks

Take-Home Message of Today

- We focus on a specific class of graph-related problems: link analysis

Take-Home Message of Today

- We focus on a specific class of graph-related problems: link analysis
- Link analysis allows to extract useful information out of the structural properties of the graph only (e.g., node's connectivity)

Take-Home Message of Today

- We focus on a specific class of graph-related problems: link analysis
- Link analysis allows to extract useful information out of the structural properties of the graph only (e.g., node's connectivity)
- Many real-world graphs (also the Web) exhibit the scale-free property

Take-Home Message of Today

- We focus on a specific class of graph-related problems: link analysis
- Link analysis allows to extract useful information out of the structural properties of the graph only (e.g., node's connectivity)
- Many real-world graphs (also the Web) exhibit the scale-free property
- Few nodes are highly connected, whilst most of them have few links

Take-Home Message of Today

- We focus on a specific class of graph-related problems: link analysis
- Link analysis allows to extract useful information out of the structural properties of the graph only (e.g., node's connectivity)
- Many real-world graphs (also the Web) exhibit the scale-free property
- Few nodes are highly connected, whilst most of them have few links
- Idea: Use node's connectivity to determine the importance of a node