Graph Analytics

Data Networks and Graphs

DATA NETWORK: set of entities together with a number of pairwise relations among them. Each entity/relation can be provided with additional information (attributes)

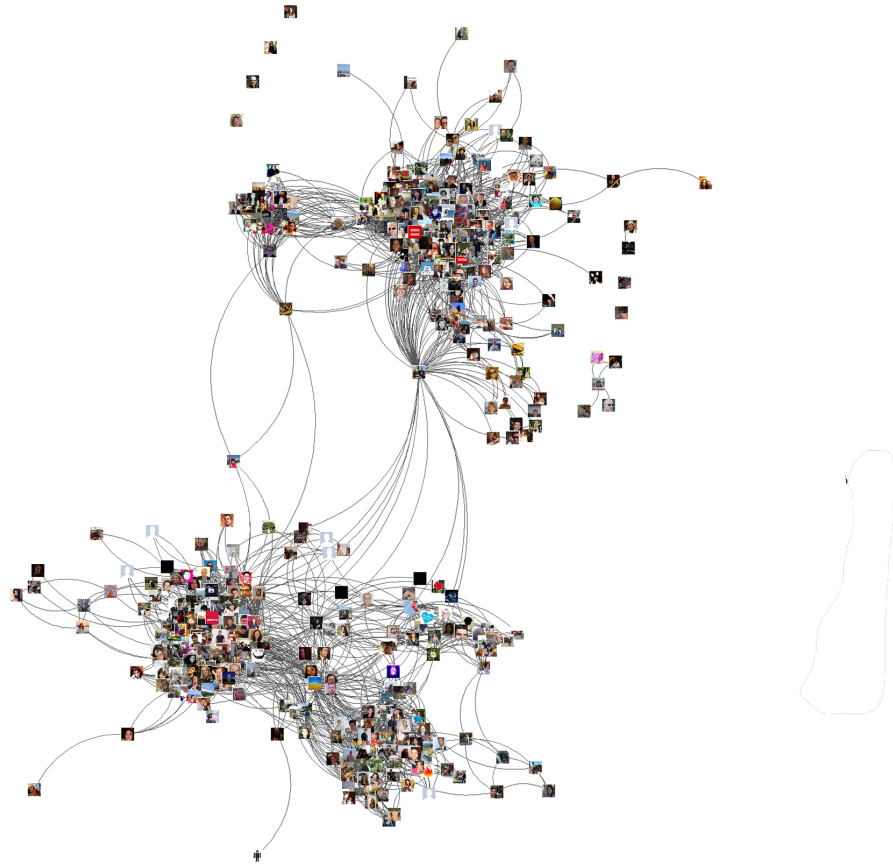
A data network can be conveniently represented as a GRAPH $G = (V, E)$

- V: (nodes) represents the entities, possibly with attributes
- $E \subseteq V \times V$: (edges) represents the pairwise relations between entities, possibly with attributes.

The graph is undirected (resp. directed) if edges are unordered (resp. ordered) pairs, and is weighted if each edge is associated with a numerical attribute representing a weight.

Examples: Social networks

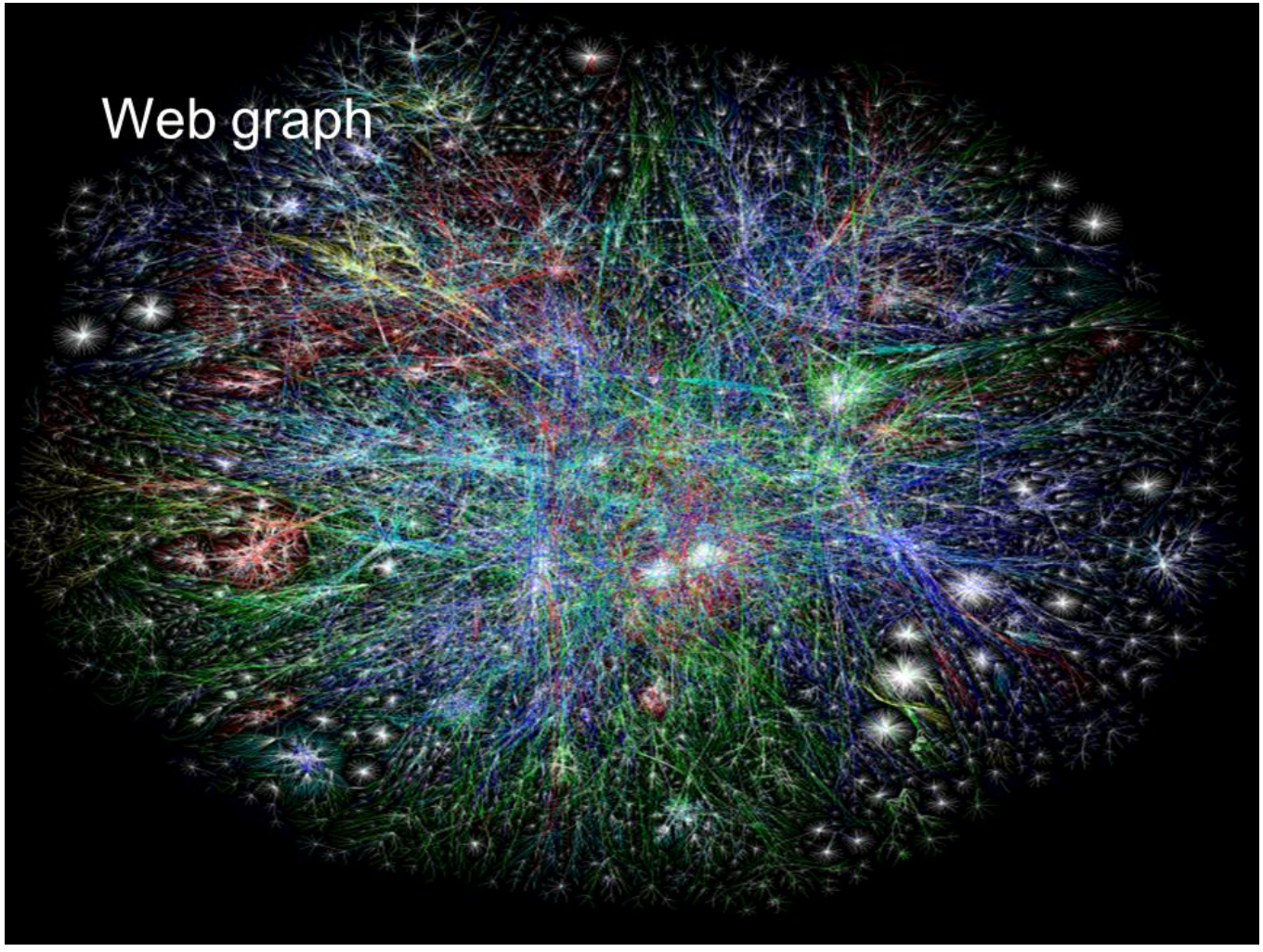
Facebook graph (about 2 billion users)



Also: Google+, LinkedIn, Twitter ..

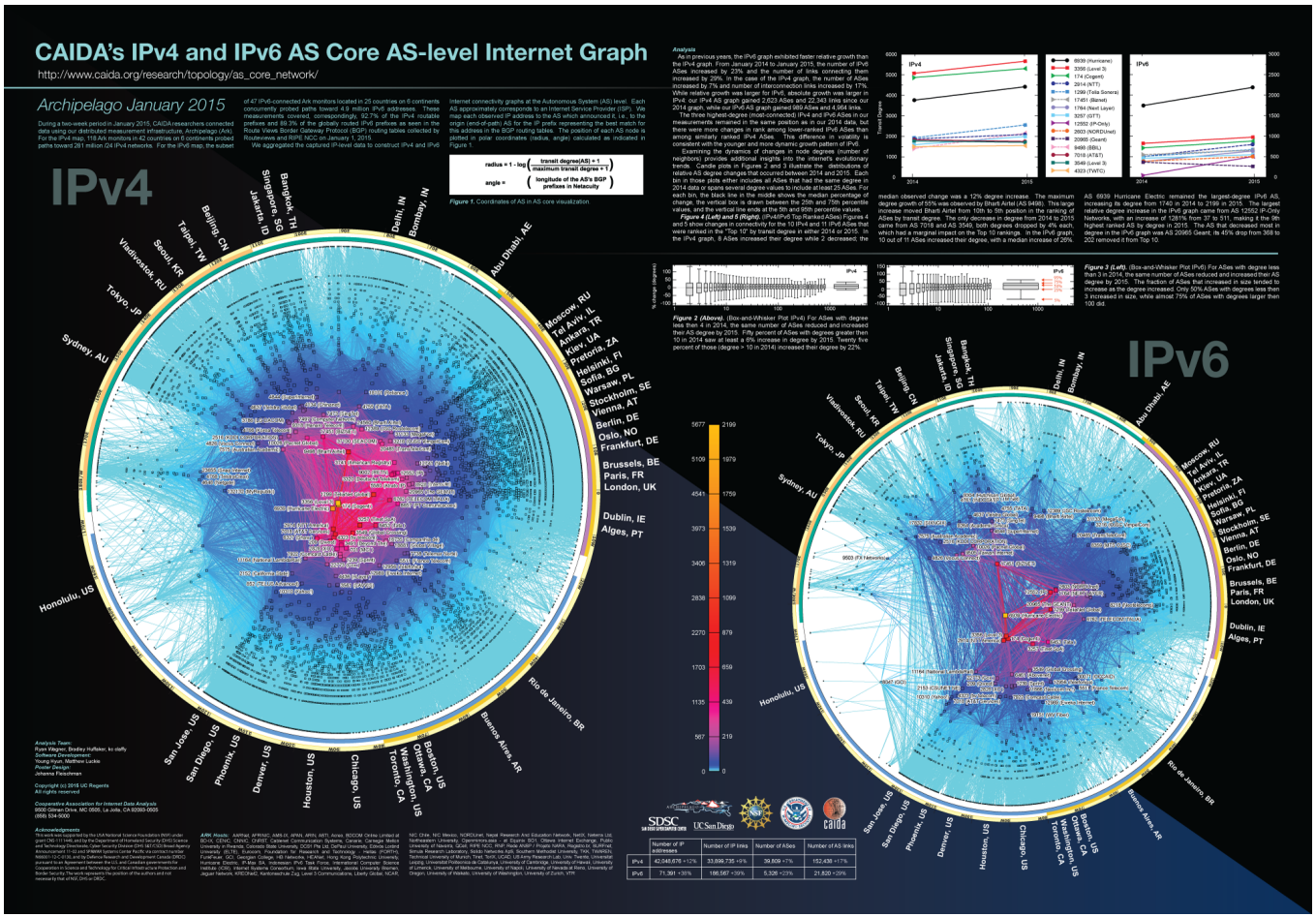
Examples: World Wide Web

Web graph (over 1 billion sites)



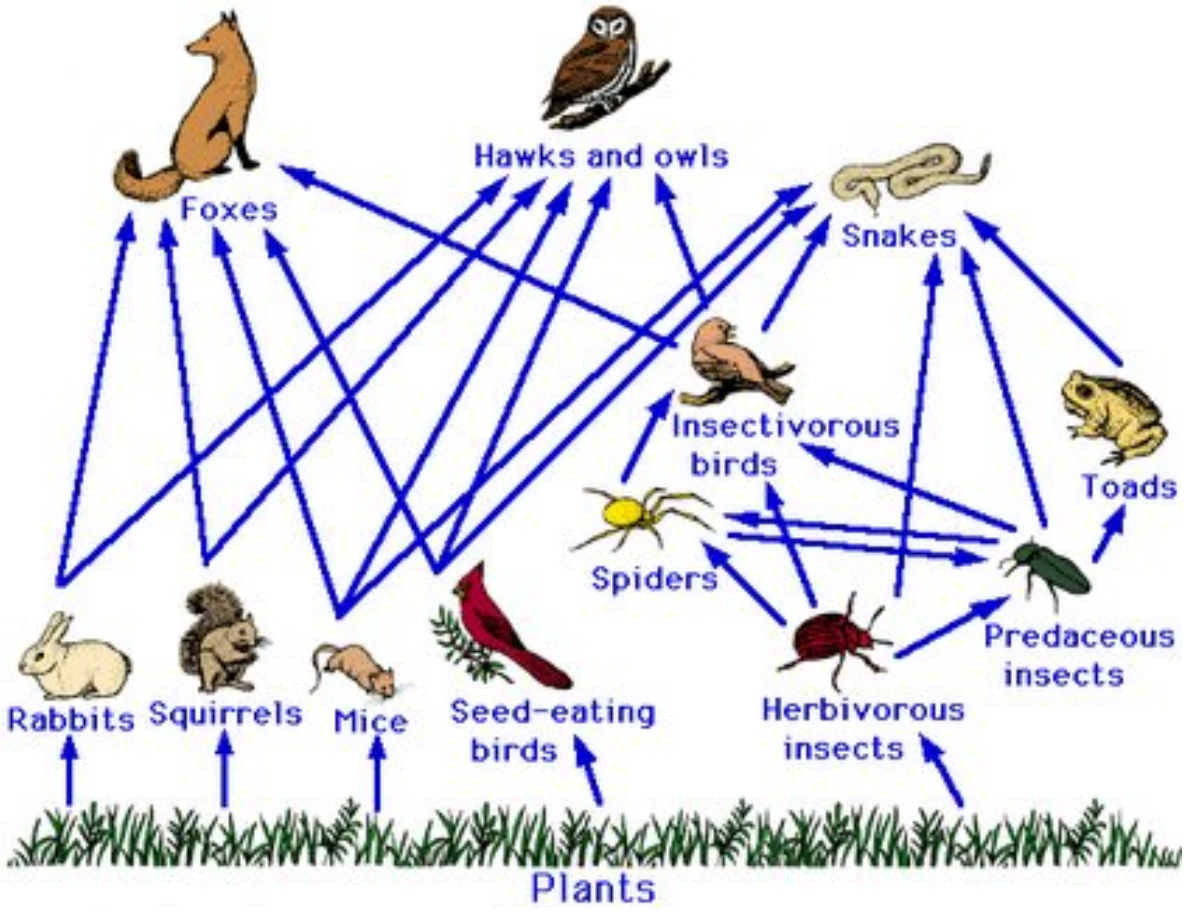
Examples: Communication networks

AS-level Internet graph (40k-50k nodes)



Examples: Biological networks

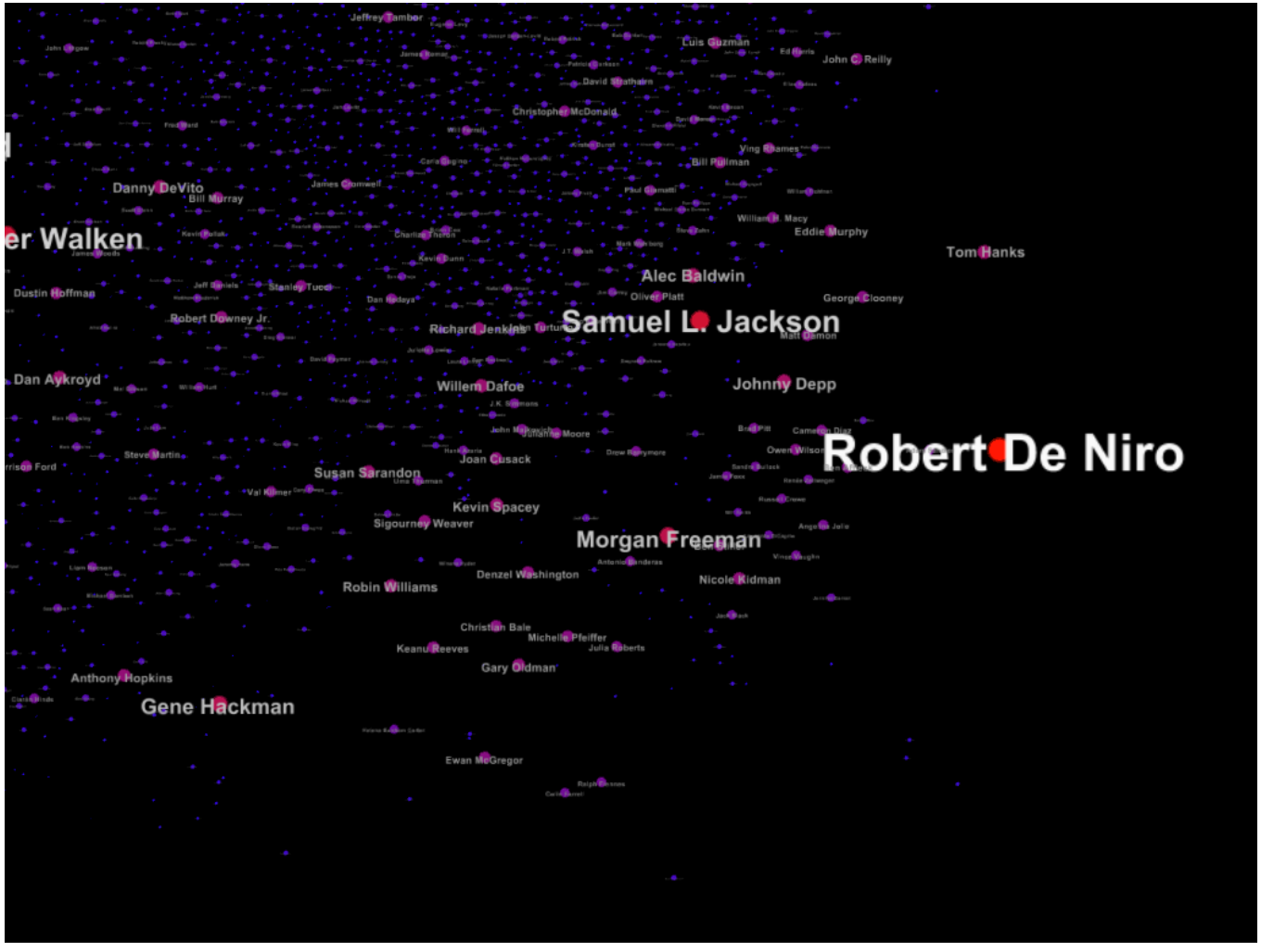
Food chain network



Also: Protein-Protein Interaction networks, biological neural networks, ..

Examples: Collaboration networks (actors)

Actors collaboration network

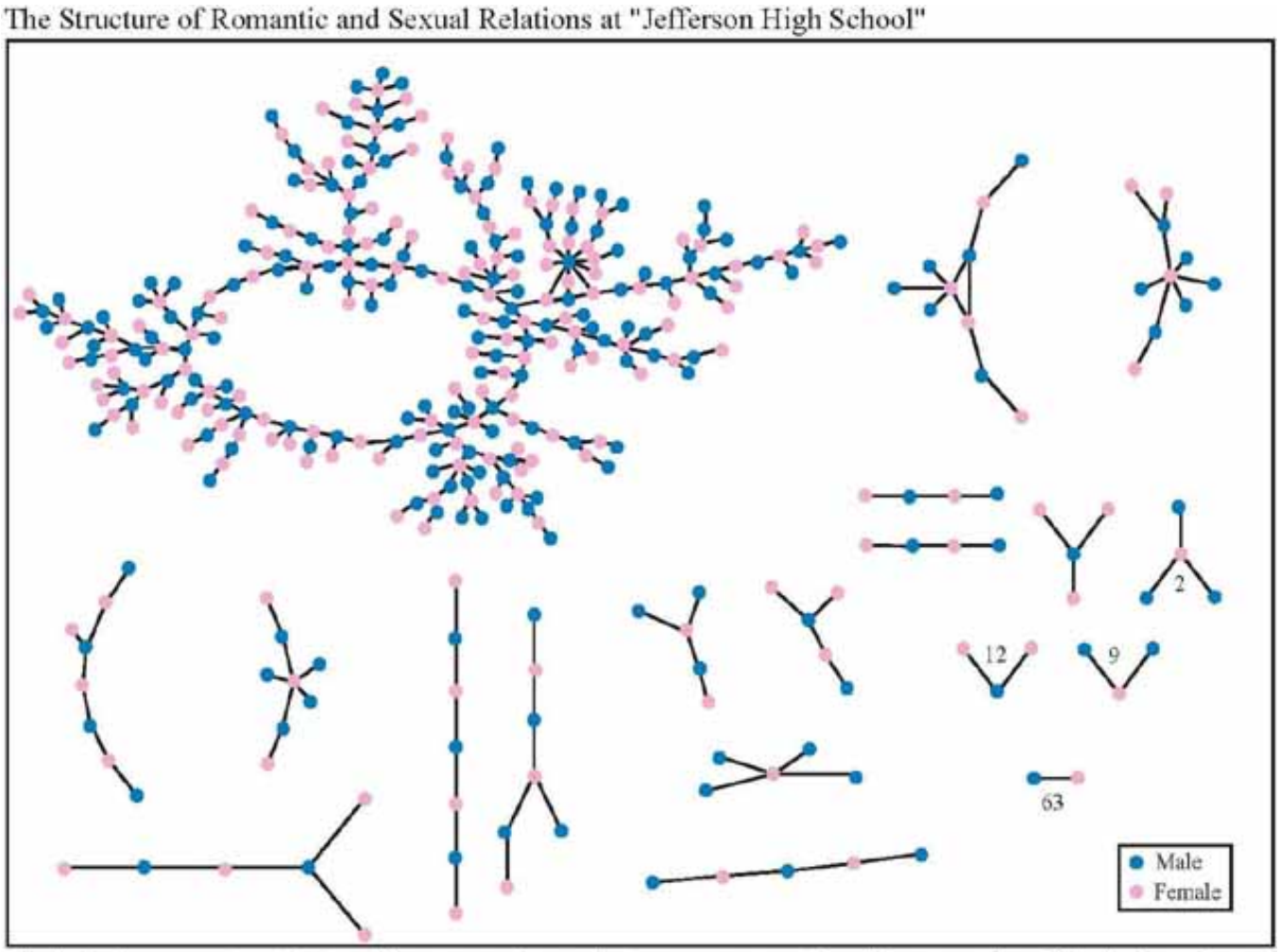


Also: co-authorship network (e.g., DBLP), citation networks

Examples: Road networks

Examples: Romantic networks

Romantic-sexual relationships at a US High School (832 students)



Each circle represents a student and lines connecting students represent romantic relations occuring within the 6 months preceding the interview. Numbers under the figure count the number of times that pattern was observed (i.e. we found 63 pairs unconnected to anyone else).

Examples: etc.

- Google+, LinkedIn, Twitter networks
- Citation networks, Co-autorship networks
- P2P networks
- Power grid networks
- Telephone networks
- Airline, railways networks
- Protein-Protein Interaction networks
- Biological neural networks
- etc.

Graph analytics: objectives

1. Connectivity analysis: e.g., s-t reachability, connected components, bridges, spanning trees/forests
2. Distance analysis: e.g., distances between nodes, diameter, radius
3. Centrality analysis: e.g., influential nodes, centralities
4. Community analysis: e.g., community detection (clustering), clustering coefficient, triangle count
5. Pattern mining: e.g., frequent subgraphs (motifs), subgraph enumeration

Remark: most problems are easily solved for small graphs (say up to 100k nodes) but become increasingly challenging, from a computational point of view, for larger graphs

Graph analysitcs: applications (some examples)

- Route/traffic optimization for logistics, distribution chains, smart cities
- Discovery of weaknesses in utility power grids or transportation networs
- Discovery of influencers or communities in a social network
- Fraud, money laundering detection in Financial Activity Networks (FANs)
- Discovery of mutation patterns associated with diseases in gene interaction networks

Review of Fundamental Primitives

Graph $G = (V, E; w)$: n nodes (set V), m edges (E), weights $w : E \to \Re$

- Breadth-First Search (BFS), Depth-First Search (DFS): systematic explorations of all nodes and edges of G.

  Sequential complexity: $O(n + m)$

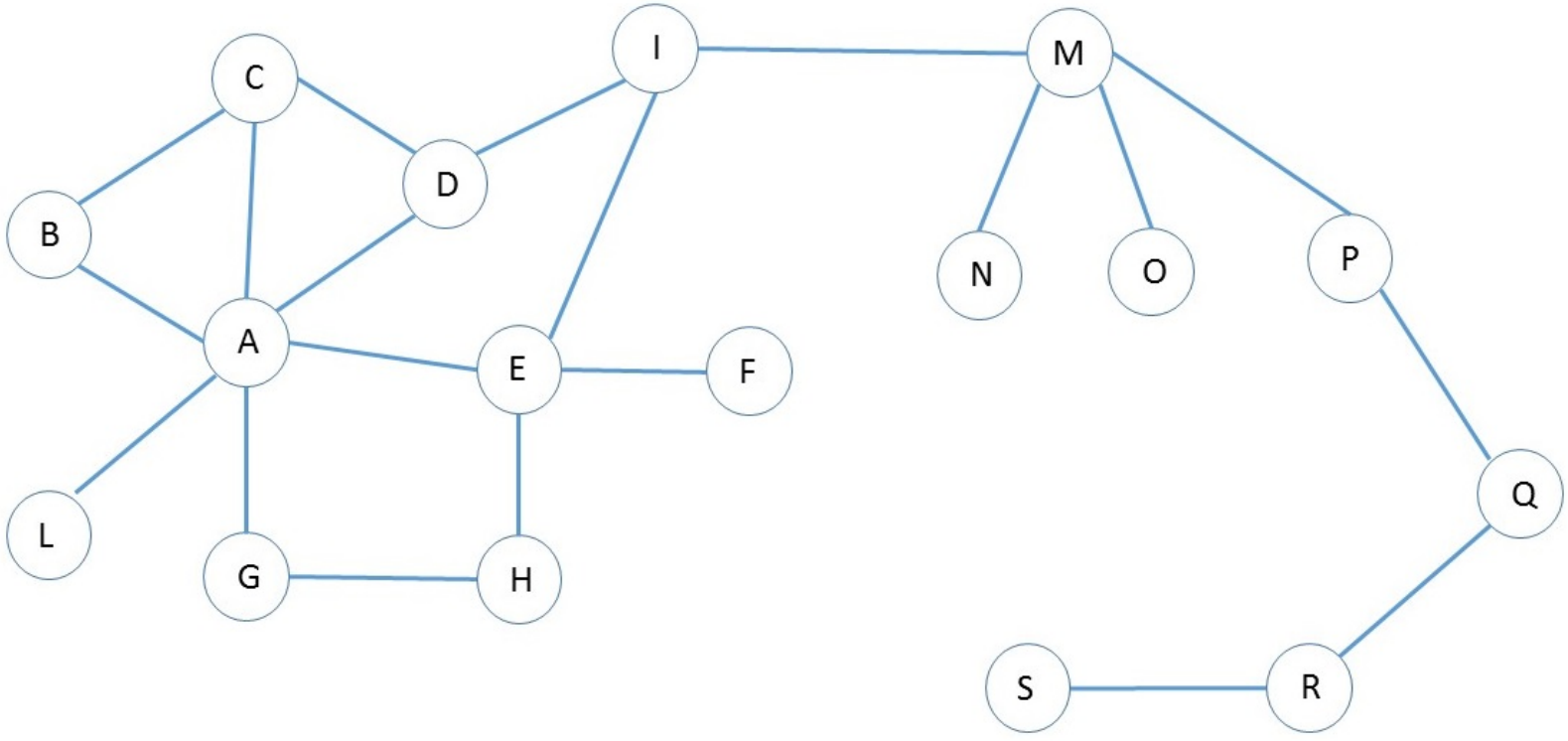
- Single Source Shortest Paths (SSSP): computation of shortest paths from a source $s \in V$ to all other nodes

  Sequential complexity:

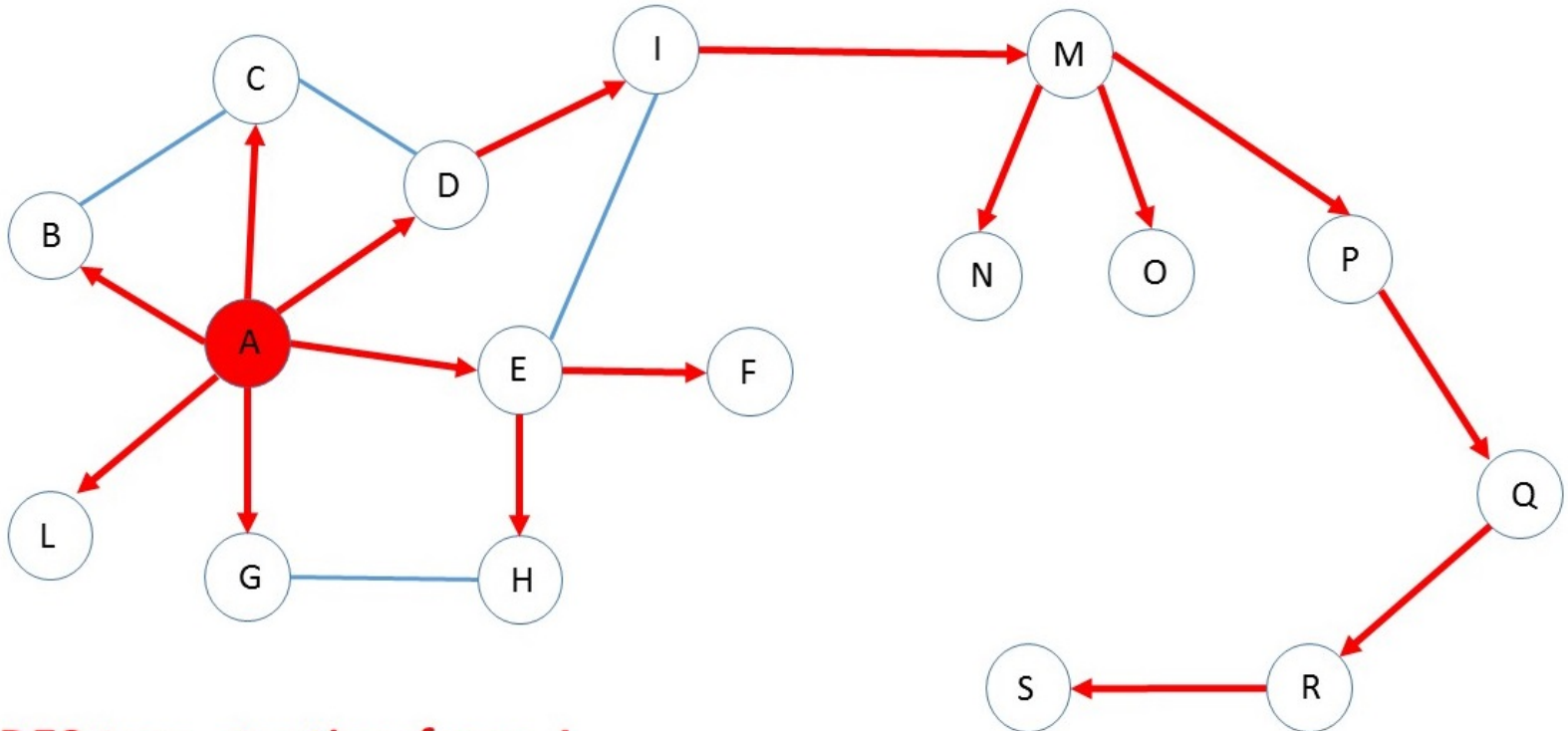
  - Unweighted (i.e., unit weights): $O(n + m)$ (via BFS)
  - Nonnegative weights: $O(n \log n + m)$ (Dijkstra)
  - Arbitrary weights (no negative cycles): $O(nm)$ (Bellman-Ford)

N.B. In what follows, we focus on undirected graphs. Adaptations to directed graphs are sometimes trivial, but other times tricky!

Example: graph



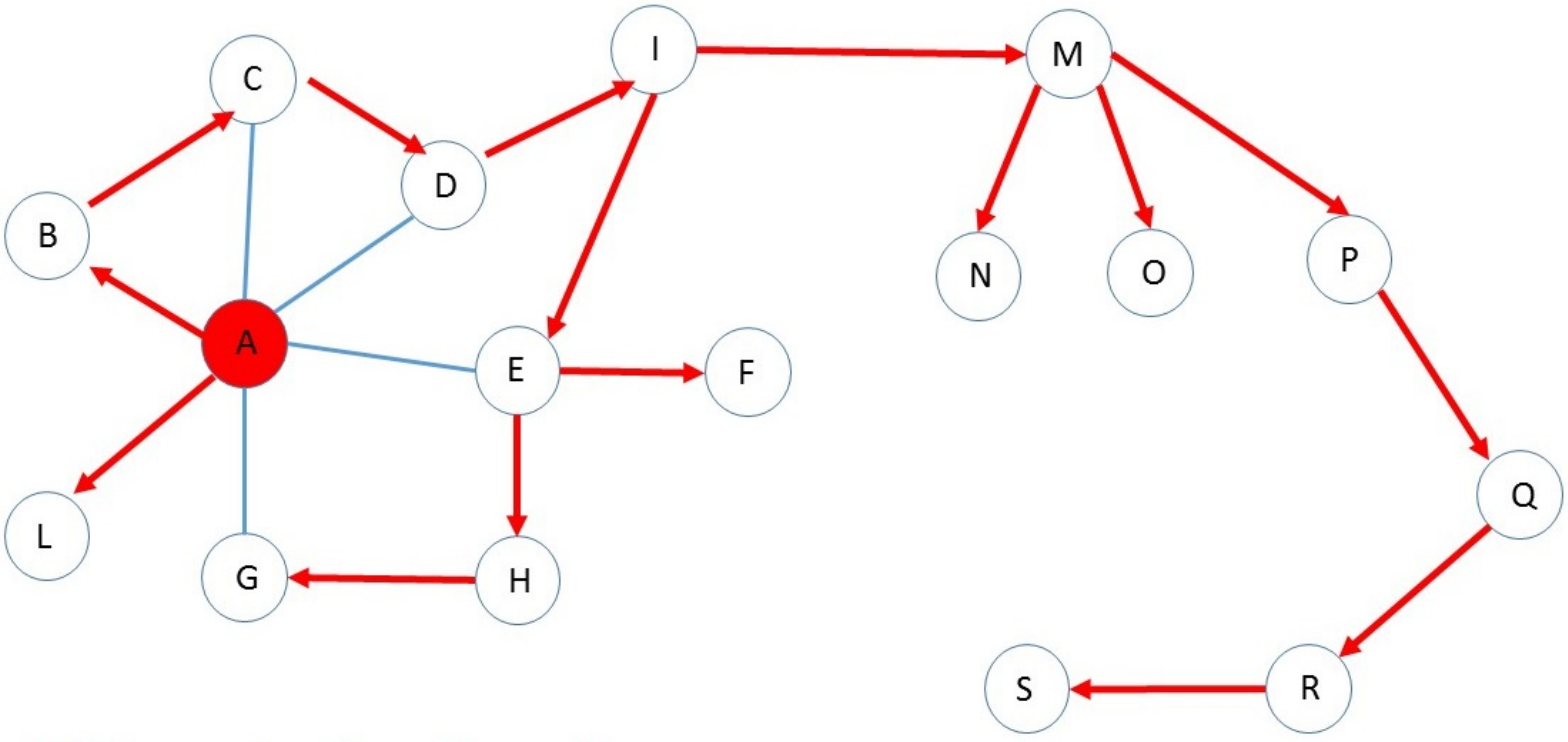
Example: BFS



BFS tree starting from A

(arrows indicate direction of discovery)

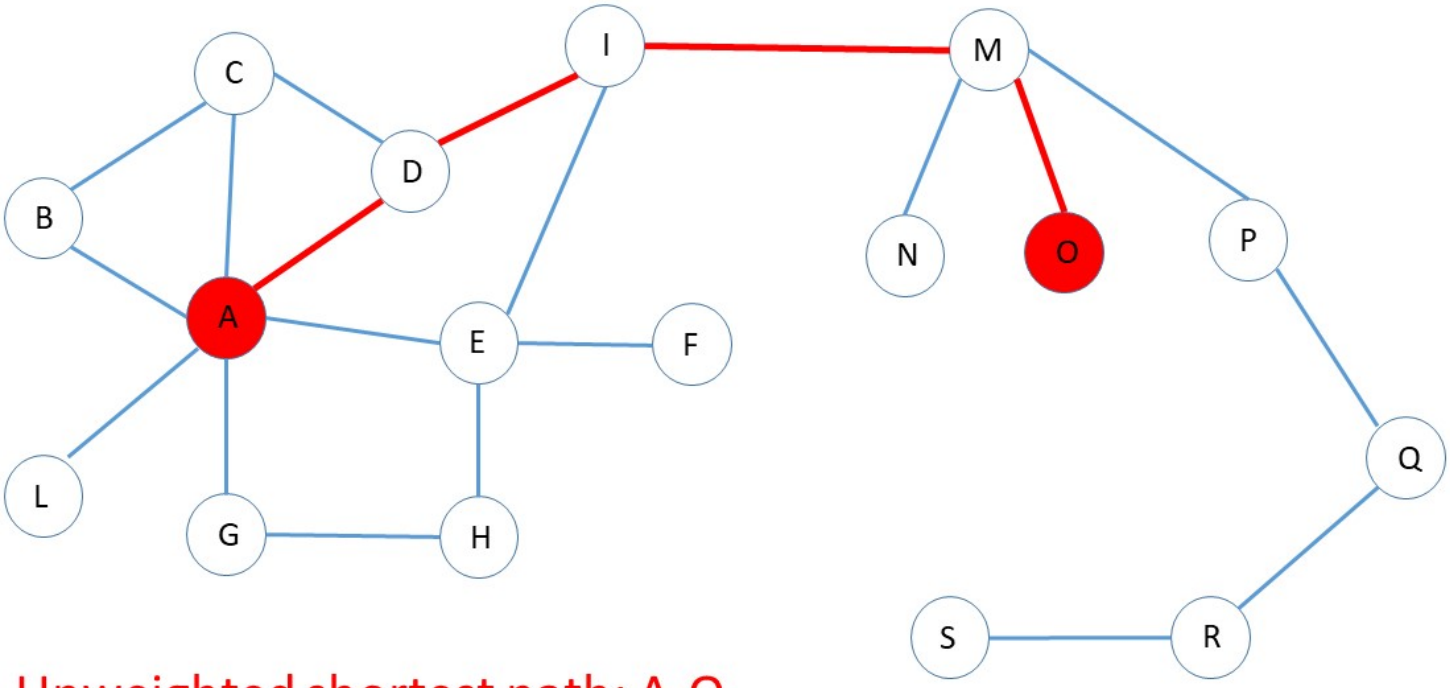
Example: DFS



DFS tree starting from A

(arrows indicate direction of discovery)

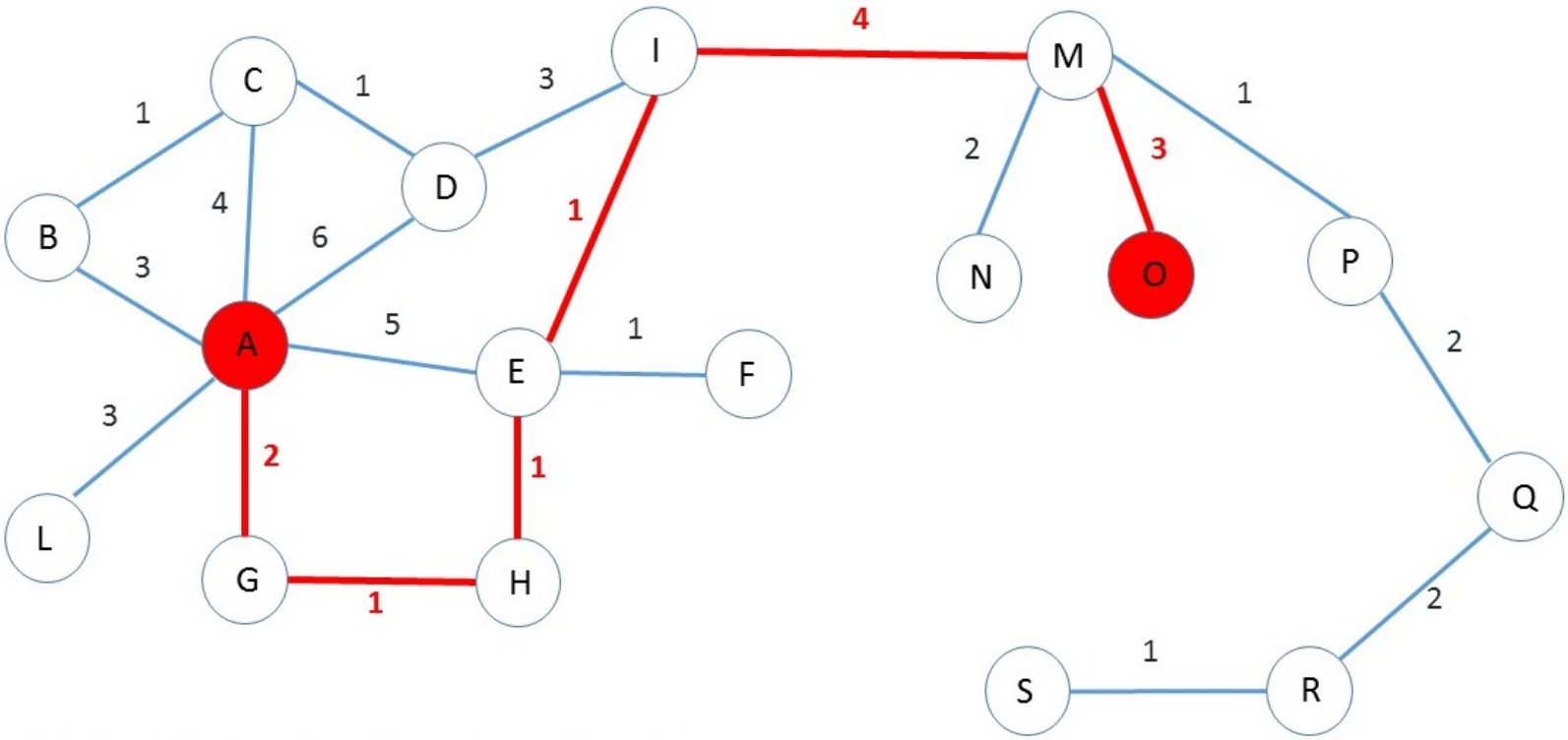
Example: shortest path (unweighted)



Unweighted shortest path: A-O

(length=4)

Example: shortest path (weighted)



Weighted shortest path: A-O

(weight=12)

Review of Fundamental Primitives (cont'd)

- Connected Components: label each $v \in V$ so to identify the maximal connected subgraph which v belgons to

  Sequential complexity: $O(n+m)$ (via BFS or DFS)

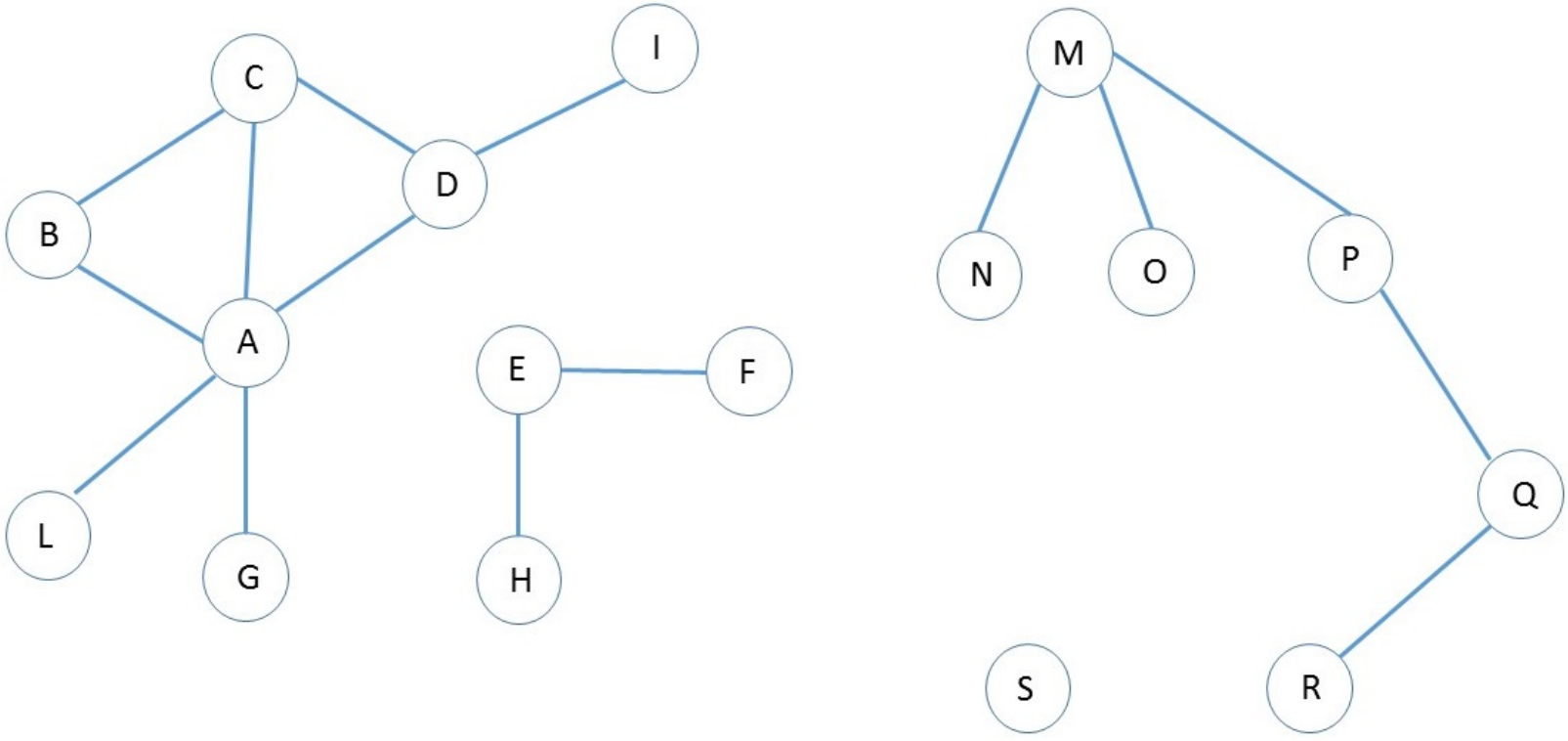
- Minimum Spanning Tree (MST) or Forest (MSF): for each connected component of G compute a spanning tree of minimum total weight[1]

  Sequential complexity: $O(n \log n + m)$ (Prim's algorithm)

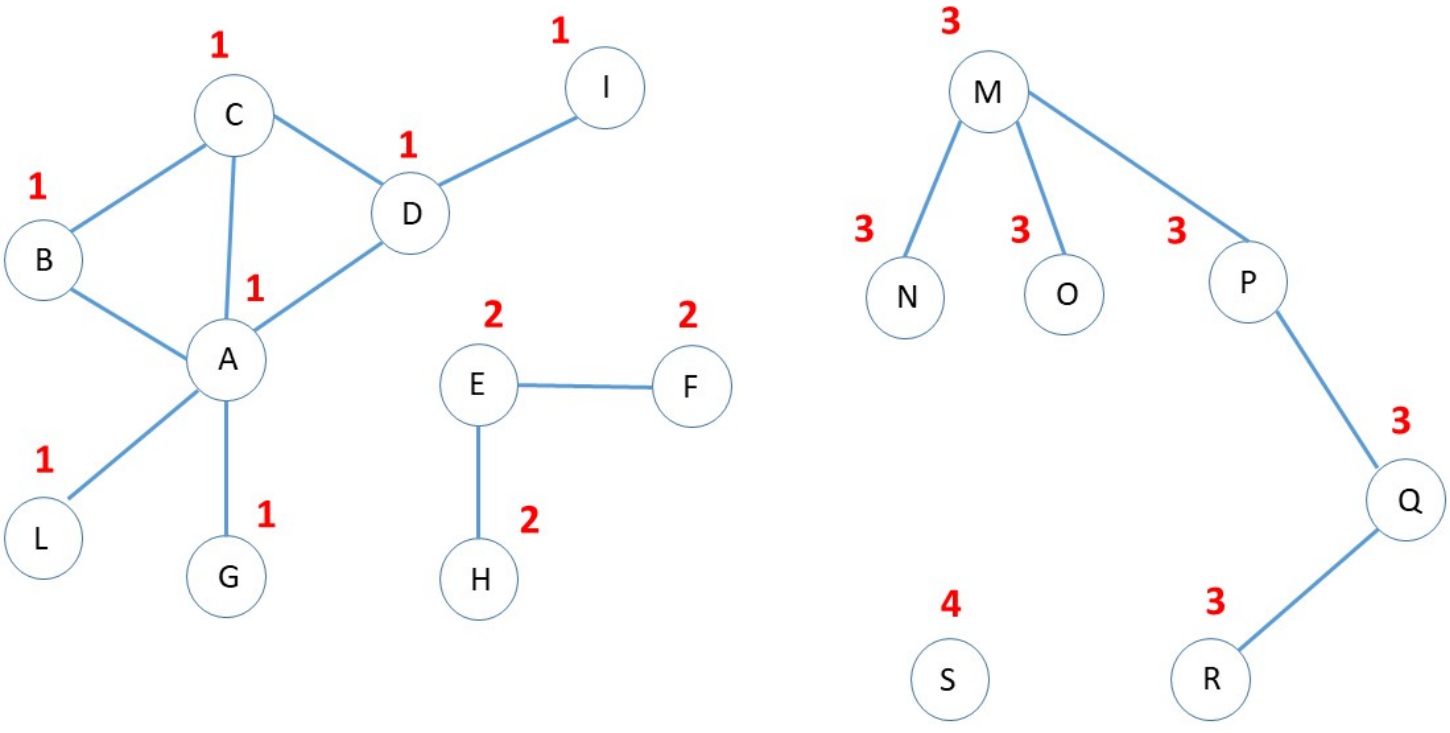
Remark: Faster MST algorithms for sparse graphs ($n \simeq m$) exist

[1] A spanning tree of a connected ν-node (sub)graph G' is a connected subgraph of G' which includes all nodes and has $\nu - 1$ edges (i.e., a tree)

Example: Connected Components

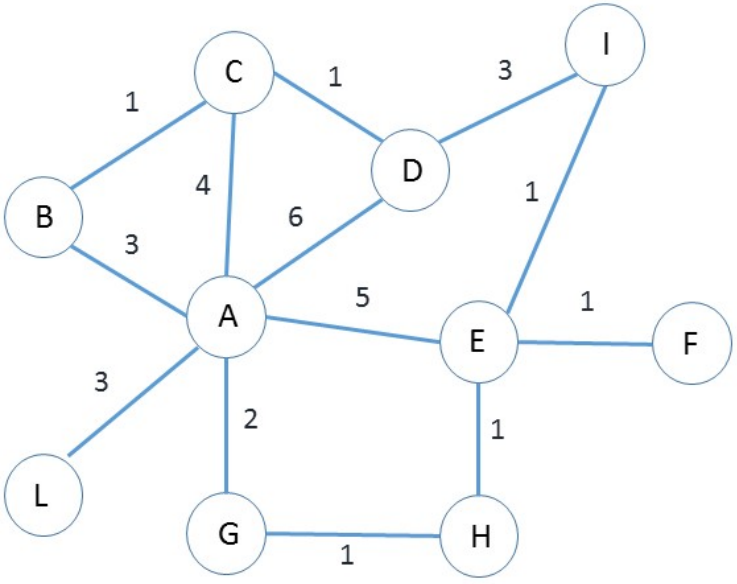


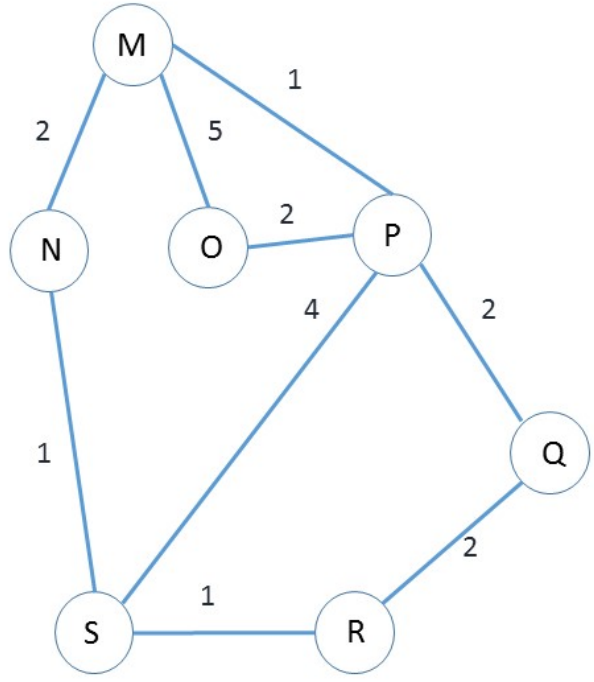
Example: Connected Components



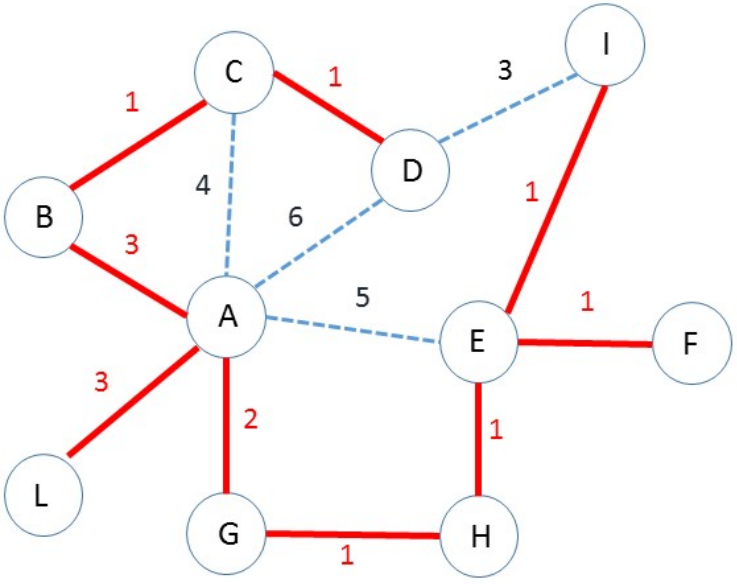
The red labels identify the Connected Components

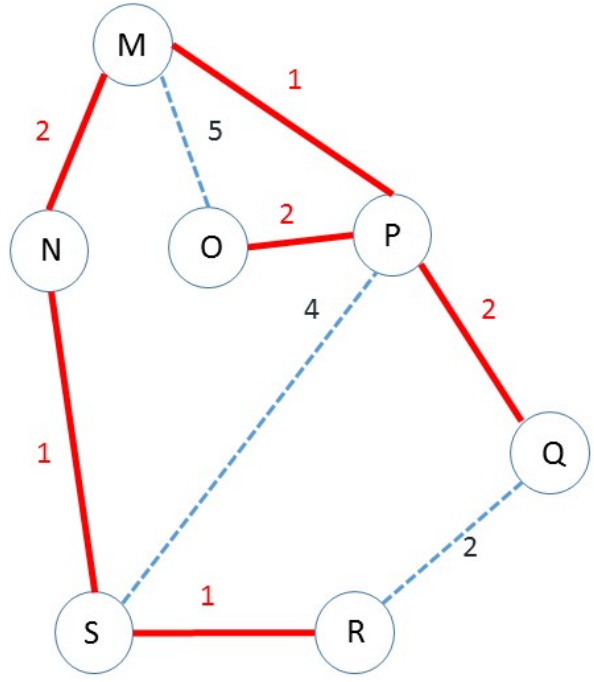
Example: Minimum Spanning Forest





Example: Minimum Spanning Forest





Minimum Spanning Forest

(total weight = 23)

Large Graph Analytics

In the big data realm, graph analytics is an area where devising algorithms that exhibit good quality guarantees and can handle efficiently large input instances is **most challenging**, for a number of reasons:

- Exact (or high-quality) solutions require superlinear number of operations.
- Distributed strategies often require several global synchronizations (hence, several MapReduce rounds) and large communication volumes which may undermine the use of low-cost cluster platforms.
- Effective sampling approaches are hard to devise (e.g., missing one edge may strongly affect certain structural properties).

Large Graph Analytics (cont'd)

We will study some prominent case studies which will allow us to introduce a variety of useful techniques.

- Minimum Spanning Forest: we will use a filtering technique which enables an efficient MapReduce solution.
- Distance-based metrics: we will focus on
  - Average Degrees of Separation;
  - Diameter;
  - Node Centralities

  which can be computed efficiently (especially on tightly-coupled platforms with large RAMs) thorugh the use of sketches or pivotal nodes.

Minimum Spanning Forest (MSF)

We now present a MapReduce algorithm (devised in [L+11]) which computes a MSF of a weighted graph G. When G is dense (i.e., m much larger than n), the algorithm uses only sublinear local space by eliminating (*filtering out*) some edges that surely do not belong to any MSF, so to make the graph small enough to be eventually processed efficiently by a single reducer

Input Weighted Graph $G = (V, E; w)$, with $|V| = n$ and $|E| = m = n^{1+c}$ for some constant $c > 0$.

Output A subset of edges defining a MSF of G, that is, a MST of each connected component of G.

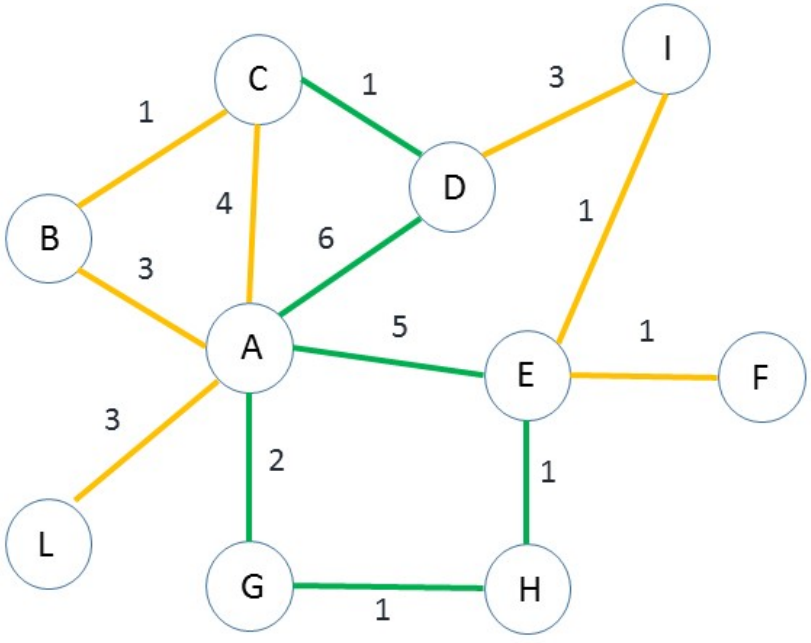
Note that the density of G is controlled by parameter c.

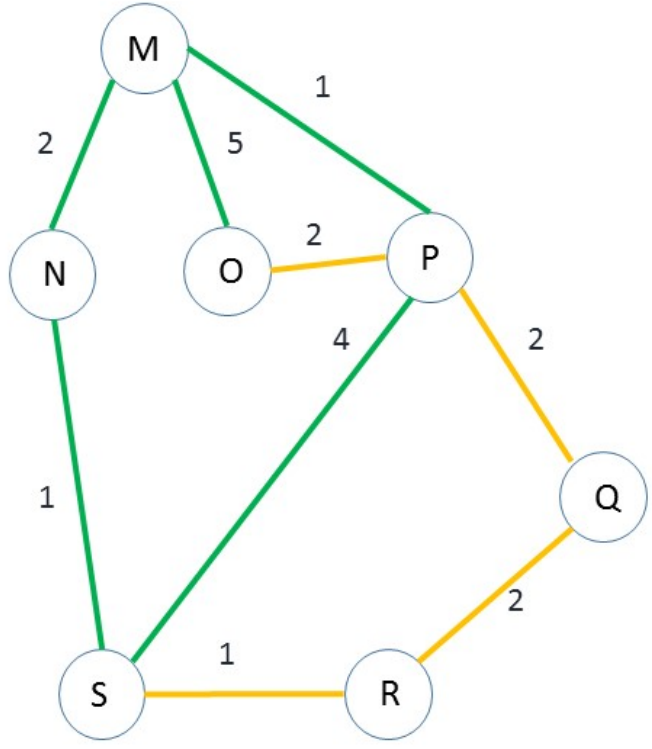
MapReduce algorithm for MSF

- **Round 1:** Partition E into $\ell = n^{c/2}$ subsets $E_1, E_2, \ldots, E_\ell$, and compute the MSF F_i of the graph induced by E_i, separately for each $1 \leq i \leq \ell$.
- **Round 2:** Compute (with a single reducer) and return the MSF of $G' = (V, F_1 \cup F_2 \cup \cdots \cup F_\ell)$, which is a subgraph of G.

Observation: it is straightforward to modify Round 2 so that the algorithm assigns labels to the nodes which identify the connected components. Therefore, the algorithm can also be used for the Connected Components primitive.

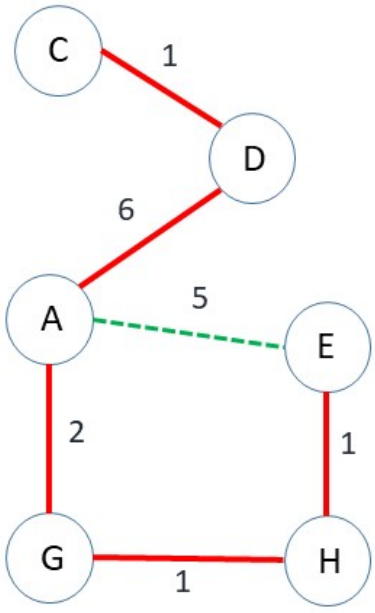
Example

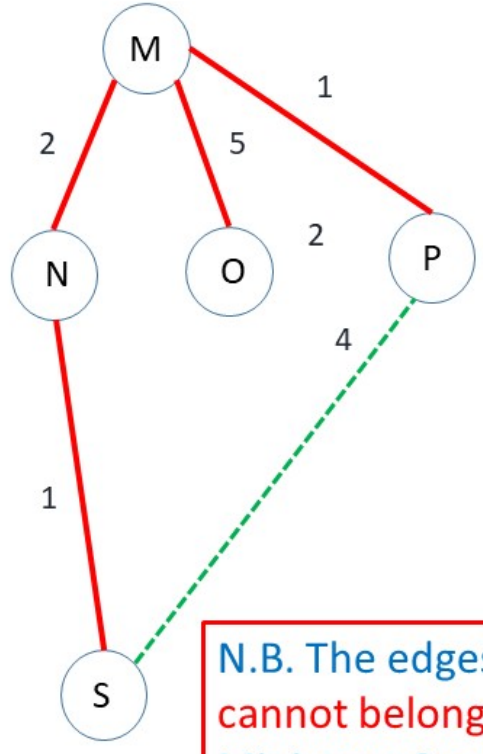




Partition into 2 subsets: E_1 E_2

Example (cont'd)

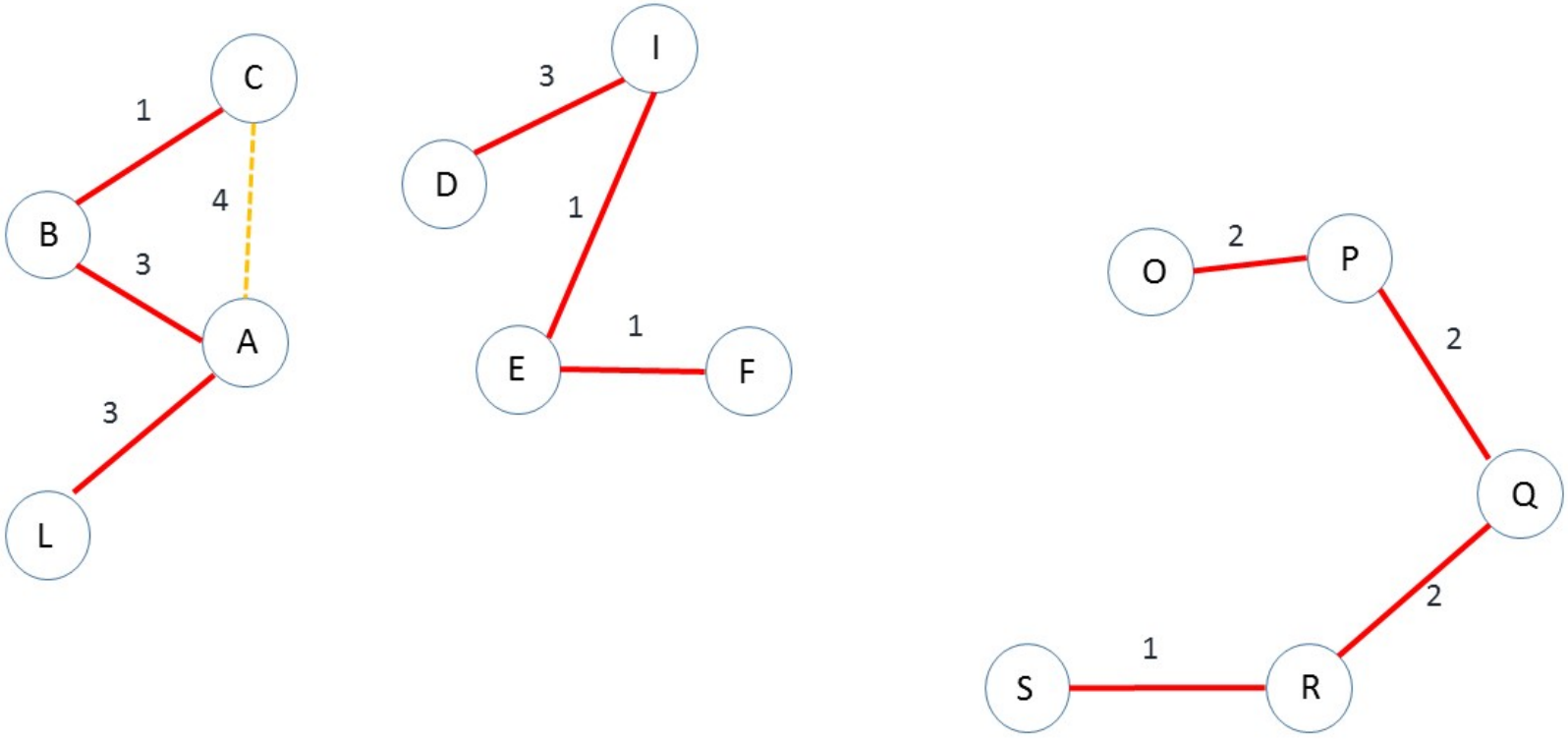




Minimum Spanning Forest for E_1

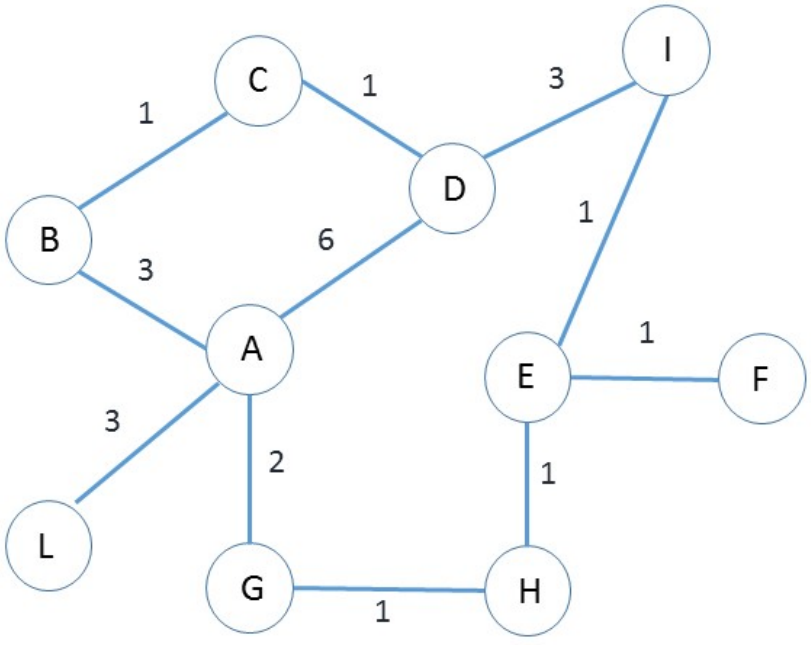
N.B. The edges filtered out cannot belong to any Minimum Spanning Forest of the original graph!

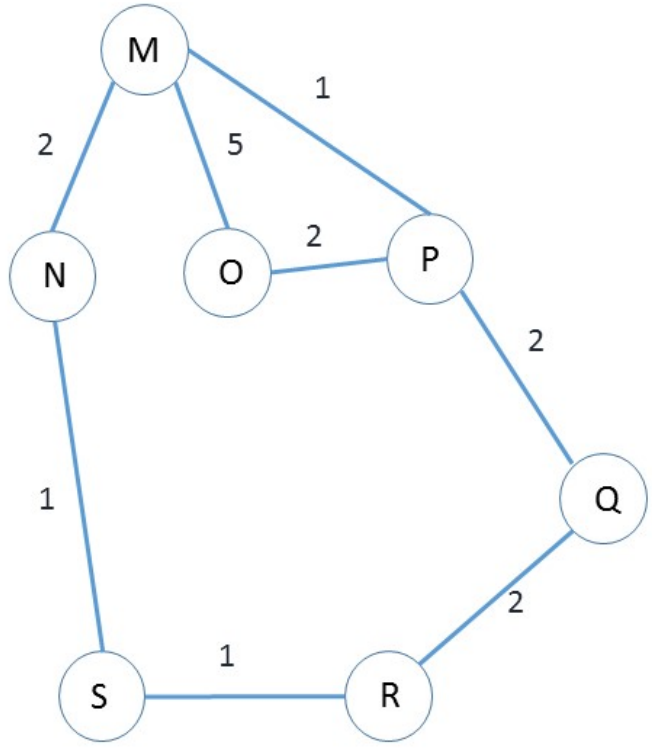
Example (cont'd)



Minimum spanning forest for E_2

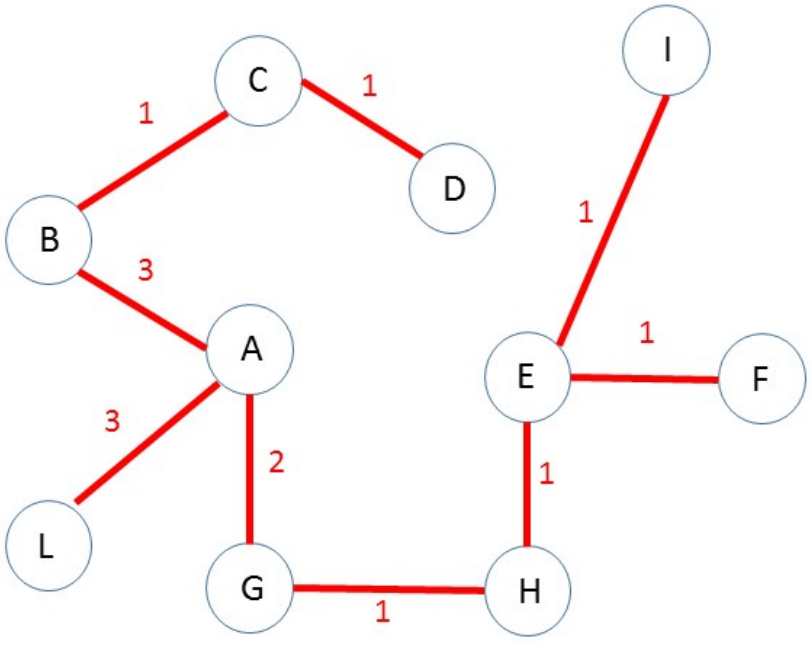
Example (cont'd)

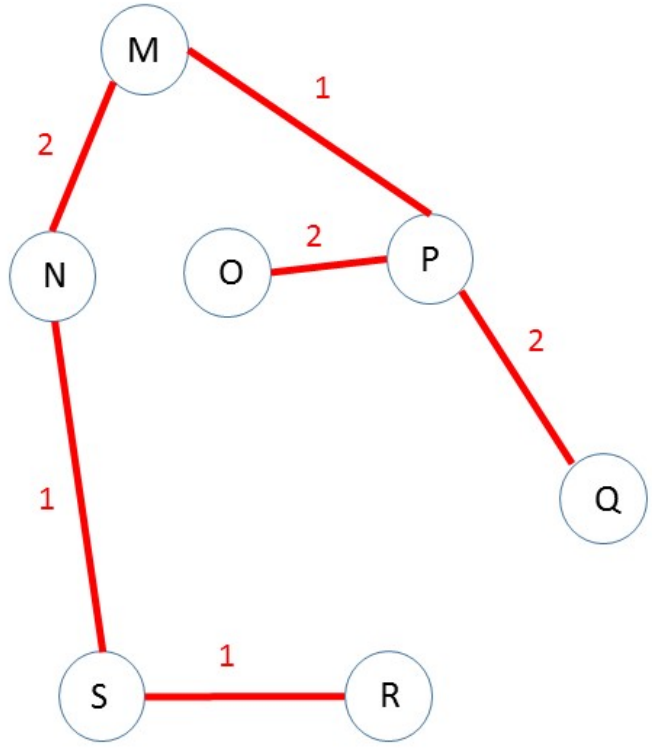




Residual graph G' after filtering

Example (cont'd)





Minimum spanning forest

MapReduce algorithm for MSF: space requirements

Basic fact: The MST of a connected graph with n nodes has *exactly* $n-1$ edges. Therefore, a MSF of a graph with n nodes, which consists of a MST for each connected component, has *at most* $n-1$ edges.

Let us now analyze the space requirements:

- In Round 1, each reducer needs local space $O(n/\ell) = O\left(n^{1+c/2}\right)$ to store an E_i and compute F_i.
- Due to the above basic fact, in Round 2, the only active reducer requires local space $O(\ell n) = O\left(n^{1+c/2}\right)$.
- No significant data replication is needed.

In conclusion:

- $M_L = \Theta\left(n^{1+c/2}\right) = o(m)$ (i.e., sublinear in the input size)
- $M_A = \Theta(m)$ (i.e., linear in the input size)

Observation: if the initial partition of E is done at random, the same local space bound holds with high probability.

MapReduce algorithm for MSF: correctness

Basic fact (*cycle property*): Let T be a MST of a weighted graph $G = (V, E)$. Adding to T any edge $(u, v) \in E - T$ creates a cycle C such that each edge $e \in C$ has a weight $w(e) \leq w(u, v)$.

Theorem

The MR algorithm for MSF is correct.

Proof

It is sufficient to show that there exists a MSF which uses none of the edges filtered out in the first round. Let H be an arbitrary MSF for G. For each edge (u, v) belonging to H with $(u, v) \in E_i - F_i$, for some i

- u and v must belong to the same tree T of the forest defined by F_i, otherwise there would be two trees for the same connected component
- By the cycle property, $T \cup \{(u, v)\}$ contains a cycle C which includes (u, v) and where (u, v) has the largest weight.

MapReduce algorithm for MSF: correctness (cont'd)

Proof of theorem (cont'd).

- Remove (u, v) from H, thus splitting some tree $T' \subseteq H$ into 2 subtrees $T'(1)$ and $T'(2)$, and replace (u, v) in H by one of the edges of C to join again the subtrees (note that an edge in C which joins $T'(1)$ and $T'(2)$ must exist!). The weight of the new spanning forest we obtain is not larger than the weight of H.

After processing each edge (u, v) of H not belonging to the F_i's as explained above, we get a new MSF using only edges of the F_i's $\square$

Trading rounds for local space

The MR algorithm for MSF can be generalized to work with $O(M)$ local space, for any $n < M < n^{1+c/2}$. Let $m_0 = m$.

- **Round 1:** Partition E into $\ell_0 = m_0/M$ subsets of size M each and compute a MSF separately for each subset. Let m_1 be the number of residual edges (i.e., edges that belong to computed forests)
- **Round 2:** Partition the residual set of m_1 edges into $\ell_1 = m_1/M$ subsets of size M each and compute a MSF separately for each subset. Let m_2 be the number of residual edges.
- ... Continue until, at the end of **Round** i, we have $m_i \leq M$. Then, in the next round (**Round** $i+1$) compute the final MSF on the m_i residual edges.

Exercise

Suppose that $M = n^{1+\epsilon}$, for some $\epsilon \in [0, c/2]$. Determine the number of rounds required by the above strategy, as a function of ϵ and c.

Distance-based Metrics:
Average Degrees of Separation and Diameter

Definitions

Consider an undirected graph $G = (V, E)$. $\forall u, v \in V$, let $\text{dist}(u, v)$ be the length of the shortest path in G between u and v. Set $\text{dist}(u, v) = \infty$ if u and v do not belong to the same connected component.

Define

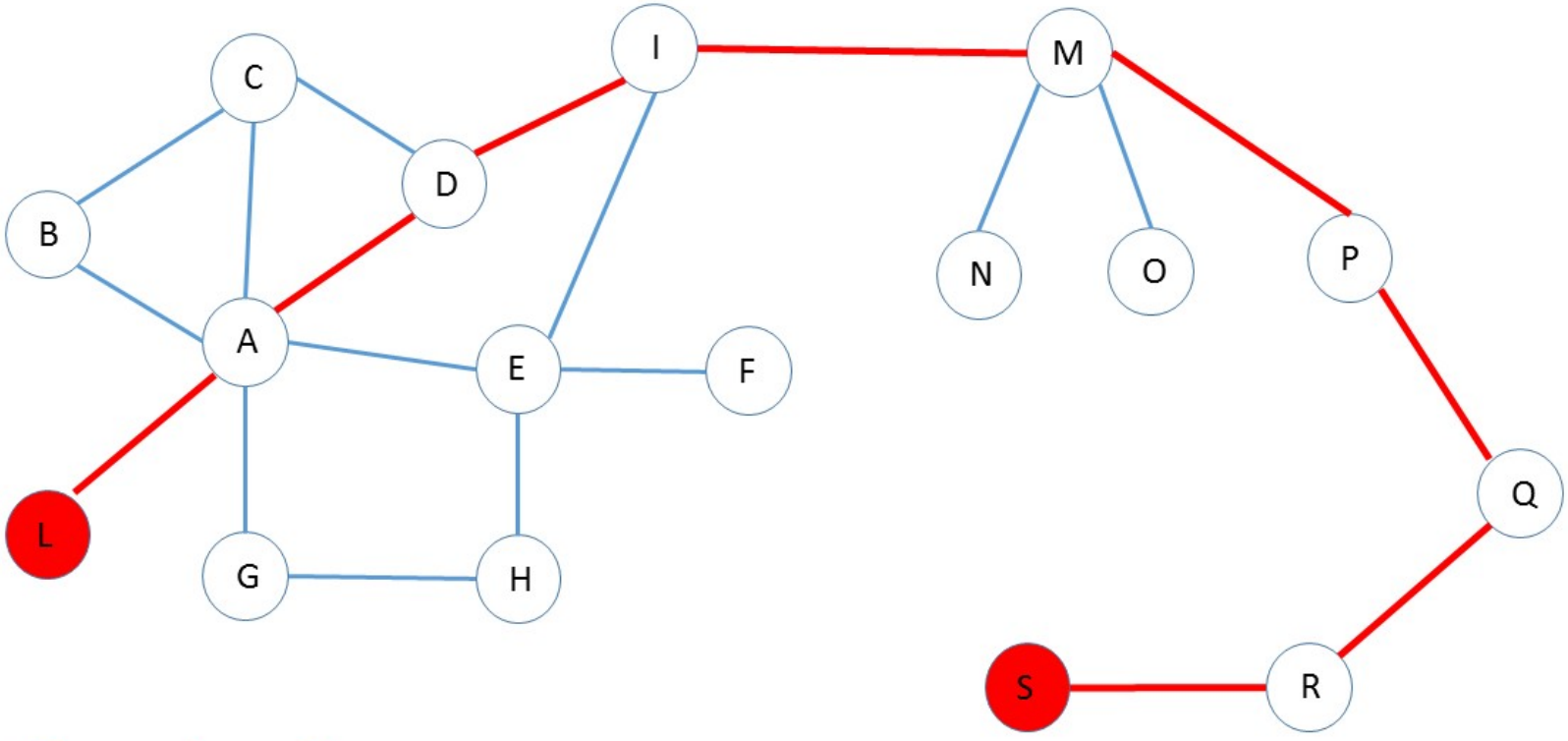
$$\text{ADS}(G) = \frac{\sum_{u \neq v \in V \,:\, \text{dist}(u,v) < \infty} (\text{dist}(u, v) - 1)}{|\{u \neq v \in V \;:\; \text{dist}(u, v) < \infty\}|}$$

$$\text{Diameter}(G) = \max\{\text{dist}(u, v) \;:\; u, v \in V \wedge \text{dist}(u, v) < \infty\},$$

where ADS stands for Average Degrees of Separation.

Remark: *We will focus on unweighted graphs where the distance between two nodes u and v is the minimum number of edges in a path from u to v. Some of the techniques we present can be extended to weighted graphs.*

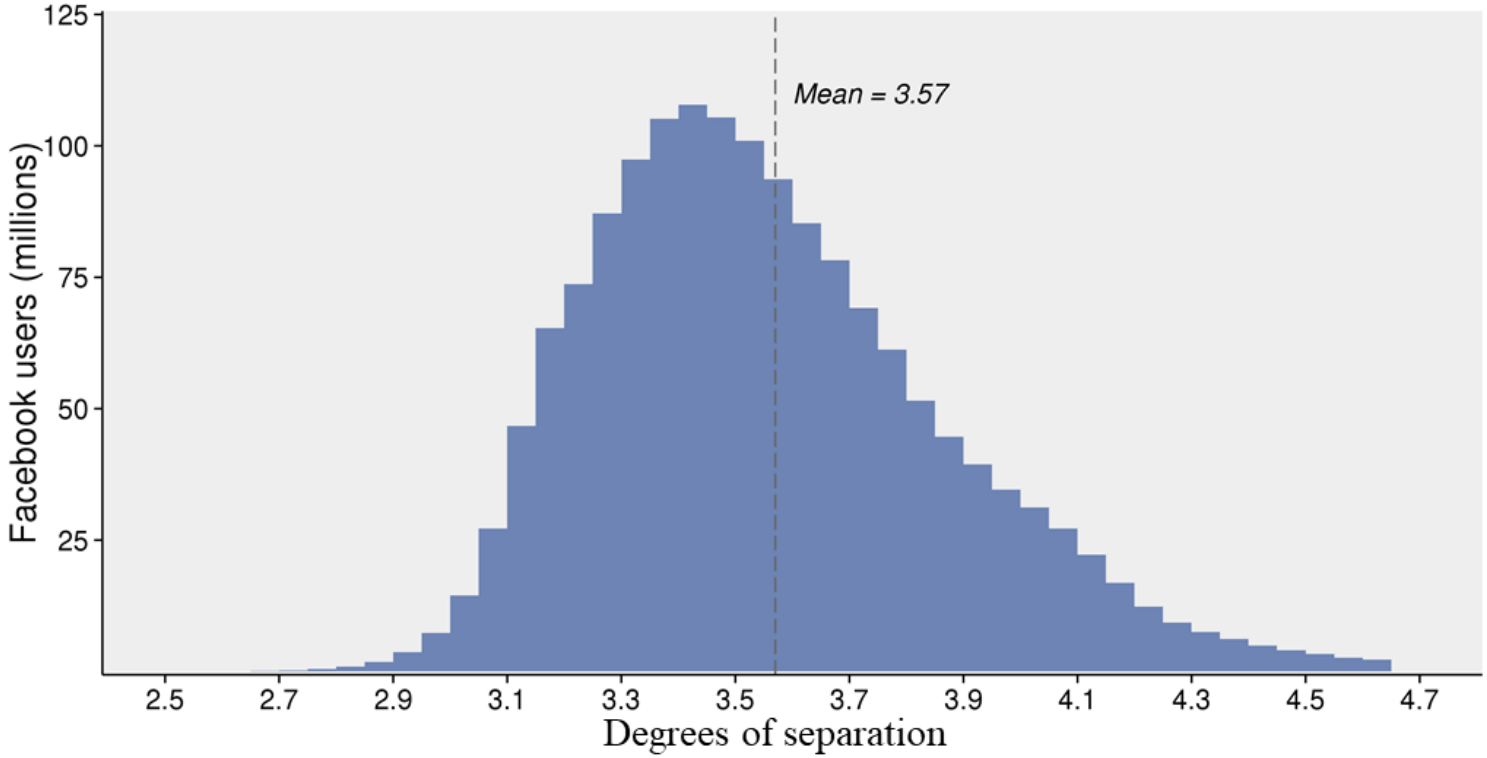
Example



Diameter = 8

Facebook graph

The following plot shows the degrees of separation (i.e., "distance"-1) between nodes in the facebook graph, as of february 2016: nodes are user profiles (about 1.6 billion at that time); undirected edges are friendship relationships [B+16].



Observation: *The graph is not connected and the mean is taken over all connected pairs*

Facebook graph: bit of history

- 1929: In a short story (*Chains*) Krigyes Karinthy conjectures that any two people are connected by a chain of at most 5 intermediaries. Hence, the theory of 6 degrees of separation (here, erroneously regarded as "distance")
- 1967: Stanley Milgram does an experimental validation of the theory: asks a group of people from Midwest (USA) to send a parcel to an unkown recipient in Massachusetts , through friends, friends of friends, etc. It discovers that the average number of intermediaries is between 4.4 and 5.7
- In 2011, Boldi et al. [B+11] analyze the FB graph (at that time with about 721 million nodes and 69 billion edges) and discover
  - Average degrees of separation: 3.74. For 92% of the connected pairs it is ≤ 4
  - Connectivity: the graph is not connected but has a giant connected component with $> 97\%$ of the nodes and diameter 41.

They made it to the New York Times (nov. 22, 2011)!

Computational issues

How easy is it to compute ADS and Diameter? Conceptually easy. Recall that the BFS from a node v provides the distance between v and all other nodes reachable from v. Thus,

Run BFS from each node

For graphs with billion nodes/edges this simple exact strategy is impractical:

- If the n BFS's sequentially, one after the other, it takes too much time.
- If the n BFS's are executed in parallel, so to reduce the running time, too much space is needed: $\Omega(n)$ space per BFS, hence $\Omega(n^2)$ space overall.

$\Rightarrow$ Resort to approximate solutions

Neighborhood function

Definition (Neighborhood function)

Let $G = (V, E)$ be an undirected, unweighted graphs. The neighborhood function $N_G(\cdot)$ of G is defined for each integer $t \geq 1$, as follows:

$$N_G(t) = |\{u, v \in V \ : \ u \neq v \wedge \text{dist}(u, v) \leq t\}|.$$

Setting $N_G(0) = 0$, we have

$$\begin{aligned} \text{ADS} &= \frac{\sum_{t \geq 1}(t-1) \cdot (N_G(t) - N_G(t-1))}{\sum_{t \geq 1}(N_G(t) - N_G(t-1))} \\ \text{Diameter} &= \min\{t \geq 1 \ : \ N_G(t) = N_G(t+1)\} \end{aligned}$$

Neighborhood Function (cont'd)

The following general strategy determines $N_G(t)$ for every t until the value stabilizes, based on the fact that if a node u is at distance $x+1>1$ from a node v then it must be at distance x from some direct neighbor of v.

for each $v \in V$ **do** $C_v \leftarrow \{v\}$

$N_G(0) \leftarrow 0$; $t \leftarrow 0$

repeat

  for each $v \in V$ **do** $C'_v \leftarrow C_v \cup (\cup_{w\,:\,(v,w)\in E} C_w)$

  for each $v \in V$ **do** $C_v \leftarrow C'_v$

  /* $\forall v$, *now* C_v *contains all nodes at distance* $\leq t+1$ *from* v, *including* v */

  $N_G(t+1) \leftarrow (1/2)\sum_{v\in V}(|C_v|-1)$ /* *each pair is counted twice in the sum* */

  $t \leftarrow t+1$

until $N_G(t) = N_G(t-1)$

return $N_G(0), N_G(1), \ldots, N_G(t-1)$

Neighborhood Function (cont'd)

Some remarks:

- Correctness easily follows from the aforementioned fact.
- ADS and diameter of G can be derived straightforwardly once the $N_G(t)$'s are known.
- A positive aspect is that in each iteration of the repeat-until loop the update of the C_v's can be parallelized.
- A negative aspect is that for each v, C_v may require space proportional to n, hence a total (aggregate) space $\Omega\left(n^2\right)$ is needed. Moreover, in an iteration, up to $\Theta(n)$ operations per node may be performed.

$\Rightarrow$ an exact implementation of the above strategy is not practical!

Algorithm HyperANF

Algorithm HyperANF, devised in [BRV11], gives an efficient approximate implemention of the general strategy for computing the neighborhood function (hence, ADS and diameter).

At the core of the algorithm is the representation of each C_v (in every iteration) through a sketch $\tilde{C}_v$ whose main features are:

- $\tilde{C}_v$ requires only $O(\log \log n)$ bits!
- An accurate estimate of $|C_v|$ is easily computed from $\tilde{C}_v$.
- The sketch of the union of two (or more) C_w's can be easily computed as a function of their sketches.

Sketch for a set S

Tool: An approximate yet accurate representation of a set S of size $O(n)$ (in our case C_v) with $O(\log \log n)$ bits can be obtained through the HyperLogLog counters by Flajolet et al.

Main observation: *the cardinality of a set of uniformly distributed random numbers is closely related to the maximum number of leading zeros in the binary representation of each number in the set.*

Definition of the sketch for S. Fix an integer constant $b > 0$ and let $Q = 2^b$. We need:

- A hash function h that maps each element of S into a random sequence of bits.
- Q counters $M[i]$, with $0 \leq i < Q$, of $O(\log \log |S|)$ bits each.

Sketch for a set S (cont'd)

The sketch for S is given by the value of the Q counters, which is obtained as follows:

`Initialize each` $M[i]$ `to 0`

for each $x \in S$ **do**

$j \leftarrow$ `integer represented by the first` b `bits of` $h(x)$

$t \leftarrow$ `number of leading 0's in the remaining bits of` $h(x)$

$M[j] \leftarrow \max\{M[j], t+1\}$

It can be proved that the size of S is well approximated by the following formula

$$|S| \simeq \alpha Q^2 \left(\sum_{j=0}^{Q-1} 2^{-M[j]} \right)^{-1} .$$

where α is a suitable constant.

Algorithm HyperANF (cont'd)

HyperANF implements the general strategy for computing the neighborhood function representing each set of neighbors C_v using a sketch $\tilde{C}_v$ defined as explained above. A few remarks:

- Given two sketches $\tilde{C}_u$ (represented by $M_u[0] \dots M_u[Q-1]$) and $\tilde{C}_w$ (represented by $M_w[0] \dots M_w[Q-1]$), the sketch for the union $C_u \cup C_w$ is easily computed as $M[0] \dots M[Q-1]$, where $M[j] = \max\{M_u[j], M_w[j]\}$ for each j.
- In [BRV11] the authors argue that with this implementation (using $Q > 64$) the neighborhood function of a graph G (hence, ADS and diameter) can be approximated accurately.
- For the diameter, HyperANF provides always a lower bound, but there is no theoretical guarantee on the approximation error.
- HyperANF cannot be (easily) generalized to work for weighted graphs where distances depend on edge weights.

Diameter approximation through pivotal nodes

Let $G = (V, E)$ be an undirected and connected graph with n nodes and m edges. We can estimate the diameter within a factor 2 in $O(n + m)$ time, as follows:

- Run BFS from an arbitrary node $v \in V$.
- Return $\Delta = \max\{\text{dist}(v, u) \ : \ u \in V\}$.

Exercise

Show that $\text{Diameter}(G) \in [\Delta, 2\Delta]$.

How hard is it to get a tighter estimate?

Theoretically, quadratic time is required to get an estimate within a factor less than 1.5. In practice, however, one can do better

Diameter approximation using pivotal nodes (cont'd)

Question: if one BFS allows us to get an estimate within a factor 2 how about if we run BFSes from several nedes?

We need to introduce two primitives. For $v \in V$

- max-dist(v): returns the node $u \in v$ with the largest shortest-path distance from v.
- middle(v, u): returns a node w in the middle of a shortest path between v and u

Observation: the nodes returned by the above primitives can be efficiently identified after building a BFS tree from v. Hence can be implemented in $O(n + m)$ time

Diameter approximation using pivotal nodes (cont'd)

4-SWEEP Algorithm

$v_1 \leftarrow$ `arbitrary node in` V

$u_1 \leftarrow$ `max-dist`(v_1) /*1st sweep*/

$z_1 \leftarrow$ `max-dist`(u_1) /*2nd sweep*/

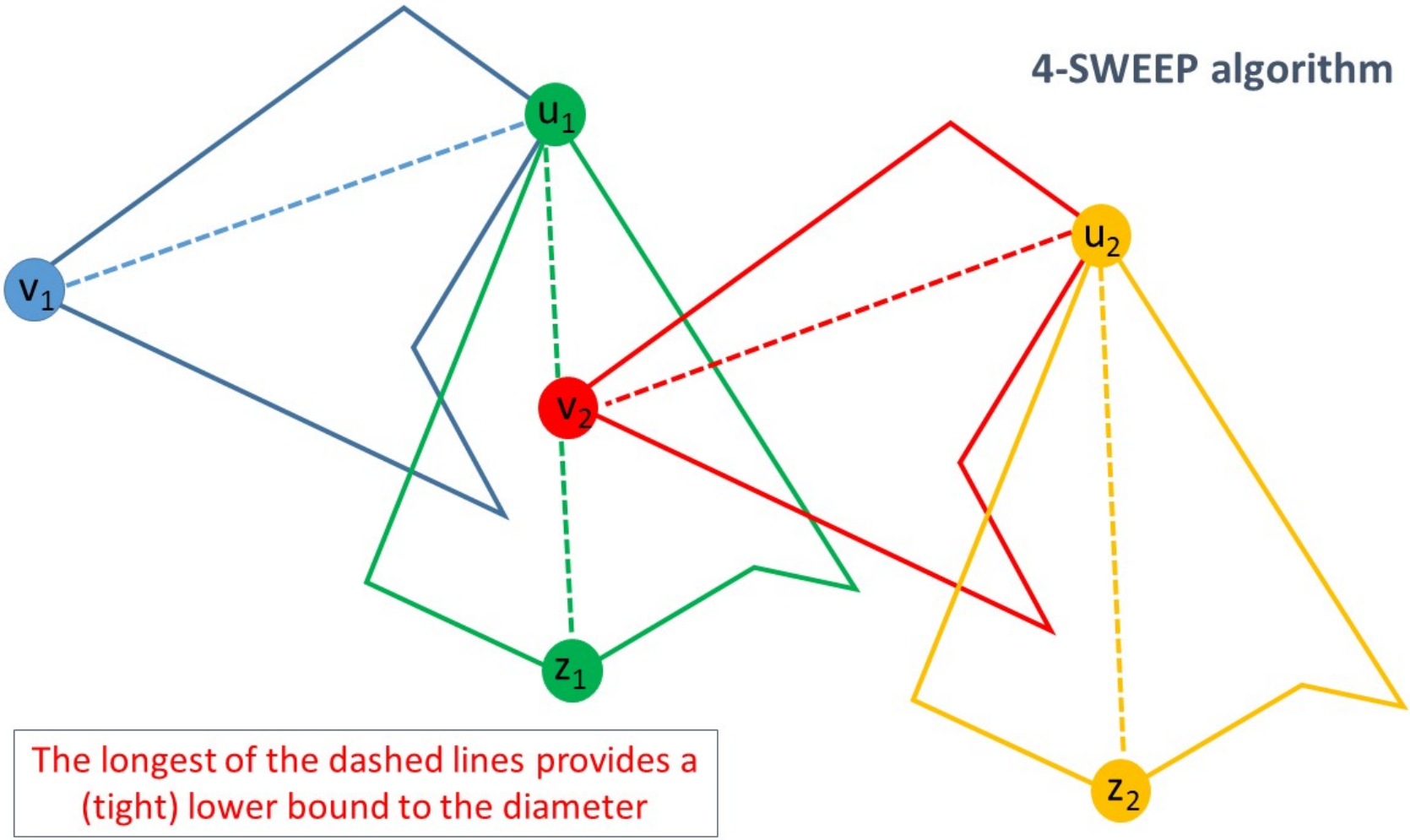
$v_2 \leftarrow$ `middle`(u_1, z_1)

$u_2 \leftarrow$ `max-dist`(v_2) /*3rd sweep*/

$z_2 \leftarrow$ `max-dist`(u_2) /*4th sweep*/

return $\Delta = \max\{\texttt{dist}(v_1, u_1), \texttt{dist}(u_1, z_1), \texttt{dist}(v_2, u_2), \texttt{dist}(u_2, z_2)\}$

Example



The longest of the dashed lines provides a (tight) lower bound to the diameter

Diameter approximation using pivotal nodes (cont'd)

- In theory we can only claim that the value Δ returned by the 4-SWEEP algorithm is such that

$$\text{Diameter}(G) \in [\Delta, 2\Delta],$$

  however, in practice Δ is very close to the actual diameter!
- If G is not connected, the 4-Sweep algorithm can be run on each connected component.
- In general, the diameter can be approximated with increasing accuracy (up to exact estimation) computing the BFS from a sufficiently large number of pivotal nodes. The exercise in the next slide explains why.
- For weighted graphs, the BFS-based diameter estimation algorithms can be employed by substituting each BFS with a SSSP computation.

Exercise

Let $G = (V, E)$ be a connected, undirected graph with n nodes. Suppose that a BFS is executed from each of $\ell > 1$ distinct pivotal nodes $v_1, v_2, \ldots, v_\ell \in V$ and that the following two values are computed:

$$\begin{aligned} R &= \max_{u \in V} \min_{1 \le i \le \ell} \mathrm{dist}(u, v_i) \\ \Delta &= \max_{1 \le i,j \le \ell} \mathrm{dist}(v_i, v_j). \end{aligned}$$

Show that $\mathrm{Diameter}(G) \in [\Delta, \Delta + 2R]$.

Observation. Note that the value R is nonincreasing with ℓ and, in fact, $R = 1$ when $\ell = n$. Therefore, one can dynamically adjust the value k until R becomes sufficiently small.

Implementation considerations

The algorithmic strategies presented in the previous slides were not targeted to a specific computational framework (e.g., MapReduce).

- Both HyperANF and BFS/SSSP-based algorithms can be efficiently implemented on multicores with a lot of shared RAM.
  - The neighborhood updates of HyperANF can easily take advantage of the available parallelism and the shared RAM allows for fast data movement.
  - BFS/SSSP are fundamental primitives, hence optimized implementations are often available.
- HyperANF has been also implemented in MapReduce using $O(\text{diameter}(G))$ rounds, sublinear local space, and slightly superlinear aggregate space $O((n+m)\log\log n)$.
- A MapReduce k-center-based diameter-approximation algorithm, which works also for weighed graphs, has been recently devloped. It needs much fewer rounds than HpyerANF or BFS-based approaches and (in practice) it is very accurate.

Centrality

From [BV14]:

Real-world complex networks [...] typically generated directly or indirectly by human activity and interaction (and, therefore, hereafter dubbed "social"), appear in a large variety of contexts and often exhibit a surprisingly similar structure. One of the most important notions that researchers have been trying to capture in such networks is node centrality: ideally, every node (often representing an individual) has some degree of influence or importance within the social domain under consideration, and one expects such importance to surface in the structure of the social network; centrality is a quantitative measure that aims at revealing the importance of a node.

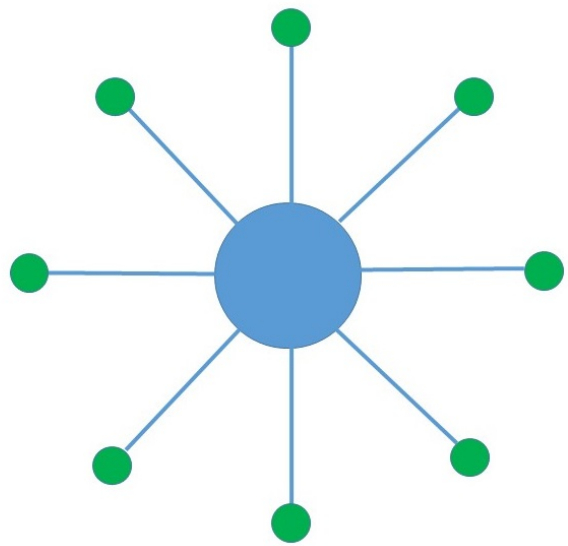
Spectrum of uses

Centrality has been used for decades (social) network analysis.

- In the late 40s, Alex Bavelas, American psychologist at MIT defined a notiuon of centrality for revealing the impact of communication patterns in a group's performance.
- Since then, centrality has also been used to, understand political integration in Indian social life, to study communication paths for urban development, to explore efficient design of organizations, to explain the wealth of the Medici family w.r.t. marriages and finantial transactions in the 15th century (see links in [BV14]).
- Centrality is also used for ranking purposes (e.g., in the context of web search).

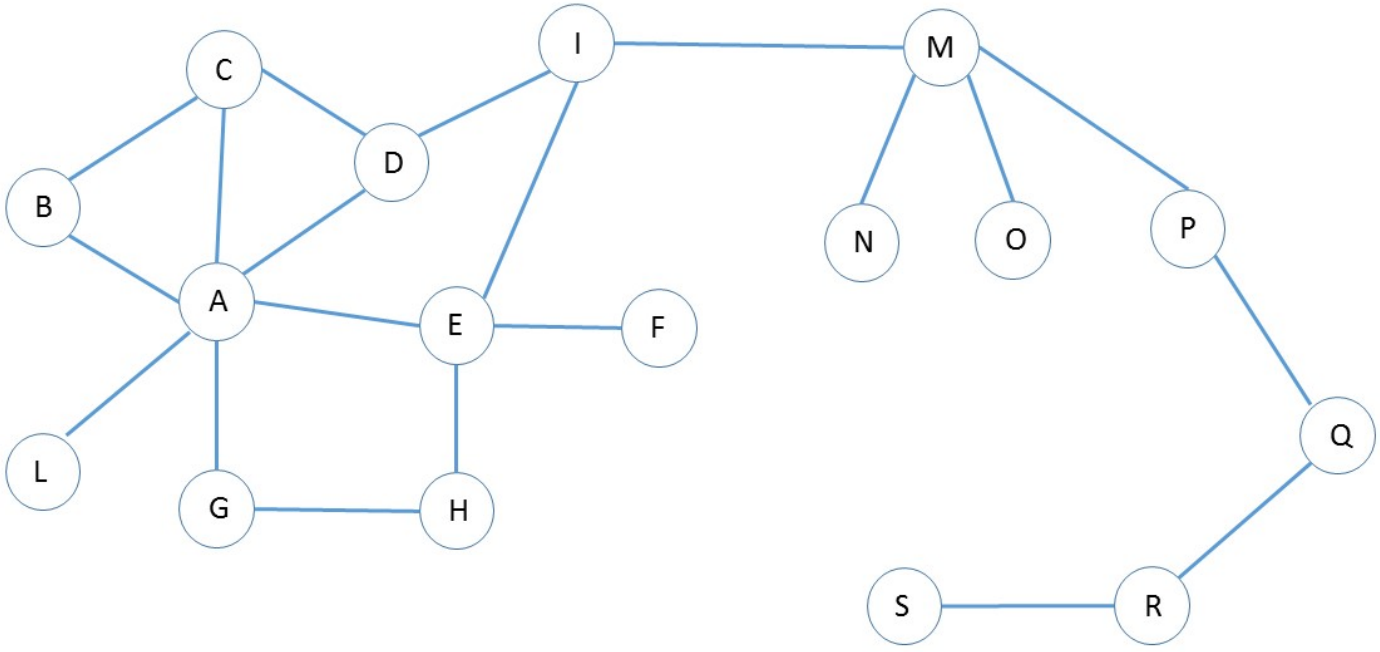
Measures of centralities

- There are many alternative measures of centrality of a node in a graph. The excellent survey [BV14] distinguishes among different types of measures: geometric, which are distance-based (e.g., *closeness*); and path-based (e.g., *betweennes*); spectral (e.g., *page rank*).
- While quite different from one another, most known definitions of centrality stem from the natural idea that a node at a center of a star is more central than the periphery



Degree centrality

Given a graph $G = (V, E)$ the simplest measure of centrality of each node $v \in V$ is its degree (if G is undirected) or in-degree (if G is directed)



Ranking by decreasing degree centrality:

A(6), E(4), M(4), C(3), D(3), I(3), B(2), G(2), H(2), P(2), Q(2), R(2), F(1), L(1), N(1), O(1), S(1)

Closeness centrality

The notion of closeness centrality is similar to the one introduced by Bavelas and is among the oldest and most popular ones.

(Undirected) graph $G = (V, E)$ with $|V| = n$ and $|E| = m$.

For each $v \in V$

$$\text{farness:} \quad f(v) = \frac{\sum_{u \in V:\, u \neq v} \text{dist}(u, v)}{n - 1}$$

$$\text{centrality:} \quad c(v) = \frac{1}{f(v)},$$

where $\text{dist}(u, v)$ denotes again the shortest path length.

Remark: *If G is not connected, $f(v) = \infty$ and $c(v) = 0$ for each node, hence the measure becomes uninteresting*

Closeness centrality (cont'd)

The following generalization (a.k.a. Lin's index) makes closeness centrality meaningful also when G is not connected:

For each $v \in V$ let $n(v) = |\{u \in V \ : \ u \neq v \wedge \mathrm{dist}(u,v) < \infty\}|$.

Then

$$\text{farness:} \quad f(v) = \frac{\sum_{u \in V \,:\, u \neq v \wedge \mathrm{dist}(u,v) < \infty} \mathrm{dist}(u,v)}{n(v)} \cdot \frac{n-1}{n(v)}$$

$$\text{centrality:} \quad c(v) = \frac{1}{f(v)}.$$

Observations:

- *If G is connected, $f(v)$ and $c(v)$ are as before ($n(v) = n - 1$)*
- *High $c(v) \Rightarrow v$ is reached by many nodes through short paths*

Closeness centrality (cont'd)

For simplicity, suppose that $G = (V, E)$ is connected

- Computing the closeness centrality $c(v)$ for a given node v can be done by running a BFS (or a SSSP in case G is weighted) from v.

  $\Rightarrow$ computing the closeness centralities of *all nodes* (e.g., to rank them) requires n BFSes!

- Eppstein and Wang's method [EW04] to efficiently estimate all centralities: pick $\ell < n$ random pivotal nodes $v_1, v_2, \ldots, v_\ell \in V$ and run BFS from each of them. Then, for each $v \in V$ compute the approximate closeness centrality

$$\tilde{c}(v) = \begin{cases} c(v) & \text{if } v = v_i, \quad 1 \le i \le \ell \\ \frac{\ell(n-1)}{n \sum_{i=1}^{\ell} \mathrm{dist}(v_i, v)} & \text{otherwise} \end{cases}$$

Closeness centrality (cont'd)

In [EW04], Eppstein and Wang show that:

1. $1/\tilde{c}(v)$ is an unbiased estimator of $1/c(v)$, that is, its expectation (over all possible selections of pivotal nodes) is $1/c(v)$.
2. For any $\epsilon \in (0,1)$, using $\ell = \Omega\left(\log n/\epsilon^2\right)$ pivotal nodes ensures that with high probability, for each node $v \in V$ the value $1/\tilde{c}(v)$ approximates $1/c(v)$ within an additive factor at most $\epsilon D(G)$, where $D(G)$ is the diameter of the graph.

Remark: Several recent works have developed more refined methods to get tighter estimates of the closeness centralities, or to identify the nodes with the top-K centralities.

Other centrality measures: harmonic centrality

Let $G = (V, E)$ be an undirected graph with n nodes.

Harmonic centrality: for a node $v \in V$, its harmonic centrality is defined as:

$$h(v) = \frac{1}{n-1} \sum_{u \in V:\, u \neq v} \frac{1}{\text{dist}(u, v)},$$

where we assume that $1/\infty = 0$.

Comment: it is a distance-based measure similar in spirit to the closeness centrality but well defined also for disconnected graphs.

Other centrality measures: betweenness centrality

Betweenness centrality: for any three nodes u, v, w, let σ_{uw} be the number of distinct shortest paths from u to w, and let $\sigma_{uw}(v)$ be the number of such paths that pass through v. The betweenness centrality of a node $v \in V$ is defined as:

$$b(v) = \sum_{u,w \in V:\ u \neq v \neq w} \frac{\sigma_{uw}(v)}{\sigma_{uw}}$$

Comment: the intuition behind this measure (which belongs to the class of path-based centrality measures, is that a node that belongs to many shortest paths is an important junction for the graph and should therefore be considered *more central*.

Other centrality measures: page rank

Consider an *abstract surfer* than navigates a directed graph G starting from an arbitrary node. Suppose that at some point the surfer is positioned at node $v \in V$. If v has no outgoing edges, then the surfer will move to a random node in V, uniformly chosen. Otherwise, the surfer will move to a neighbor along any of the $d_v > 0$ outgoing edges, with probability $\alpha \cdot 1/n_v$, or to an arbitrary node in V, uniformly chosen, with probability $1 - \alpha$, for some $\alpha \in (0, 1)$.

Page rank: the page rank of a node $v \in V$ is the probability, at steady state, that the surfer is in node v.

Comment: This measure was originally proposed to rank web pages and captures the idea that an important (i.e., central) page should attract more the interest of a generic surfer. The factor α is called damping factor and represents the chance that the web surfer stops following hyperlinks and decides to restart navigation from a random node.

Exercise

Let $G = (V, E)$ be a connected, undirected graph with n nodes and let ℓ be an integer, $1 \leq \ell < n$. For an arbitrary $v \in V$ consider its approximated closeness centrality $\tilde{c}(v)$ computed with the Eppstein and Wang's method, using ℓ random pivotal nodes $v_1, v_2, \ldots, v_\ell$ drawn from V independently, with replacement and with uniform probability. Show that $1/\tilde{c}(v)$ is an unbiased estimator of $1/c(v)$, i.e.,

$$\mathrm{E}\left[\frac{1}{\tilde{c}(v)}\right] = \frac{1}{c(v)},$$

where $c(v)$ is the exact closeness centrality of v. For simplicity assume that $\tilde{c}(v)$ is computed as $\ell(n-1)/(n\sum_{i=1}^{\ell} \mathrm{dist}(v_i, v))$ even if v is a pivot.

Theory questions

- Consider a weighted connected graph $G = (V, E)$ and let C be a cycle in G. Let $e \in C$ be an edge with maximum weight among the edges of C. Argue that there exists a MST of G which does not contain edge e.
- Let G be a connected undirected graph. Define the diameter(G) and show that this measure can be approximated within a factor 2 in linear time.
- What are the characteristics of nodes with high closeness centrality in a graph G?

References

L+11 S. Lattanzi et al. Filtering: a method for solving graph problems in MapReduce. ACM-SPAA 2011: 85-94

B+16 S. Bhagat et al. Three and a half degrees of separation. https://research.fb.com/three-and-a-half-degrees-of-separation/

BRV11 P. Boldi, M. Rosa, S. Vigna. HyperANF: approximating the neighbourhood function of very large graphs on a budget. WWW 2011: 625-634.

BV14 P. Boldi, S. Vigna. Axioms for Centrality. Internet Mathematics 10(3-4): 222-262 (2014)

EW04 D. Eppstein, J. Wang: Fast Approximation of Centrality. J. Graph Algorithms Appl. 8: 39-45 (2004)