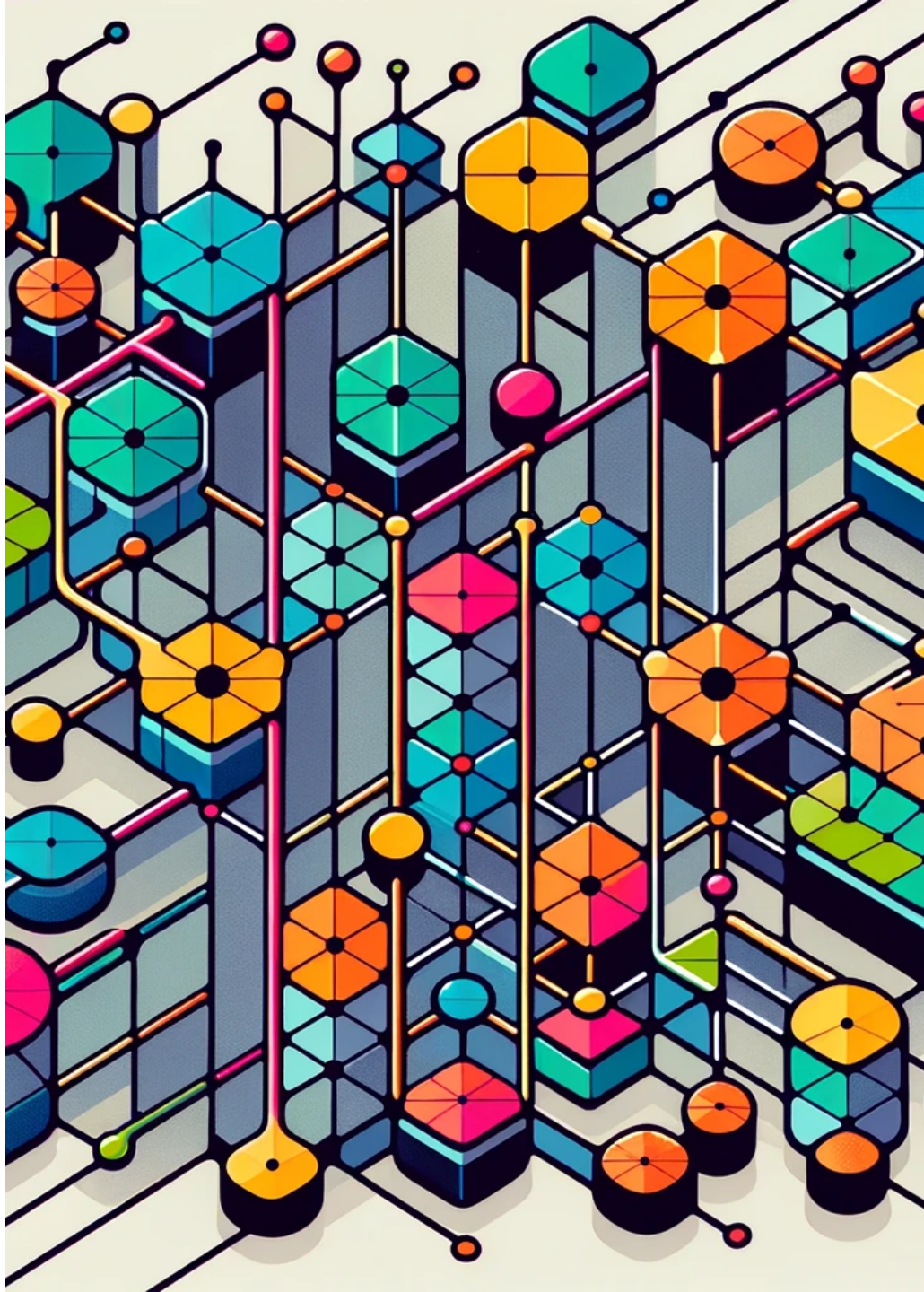
UNIVERSITÀ DI TORINO

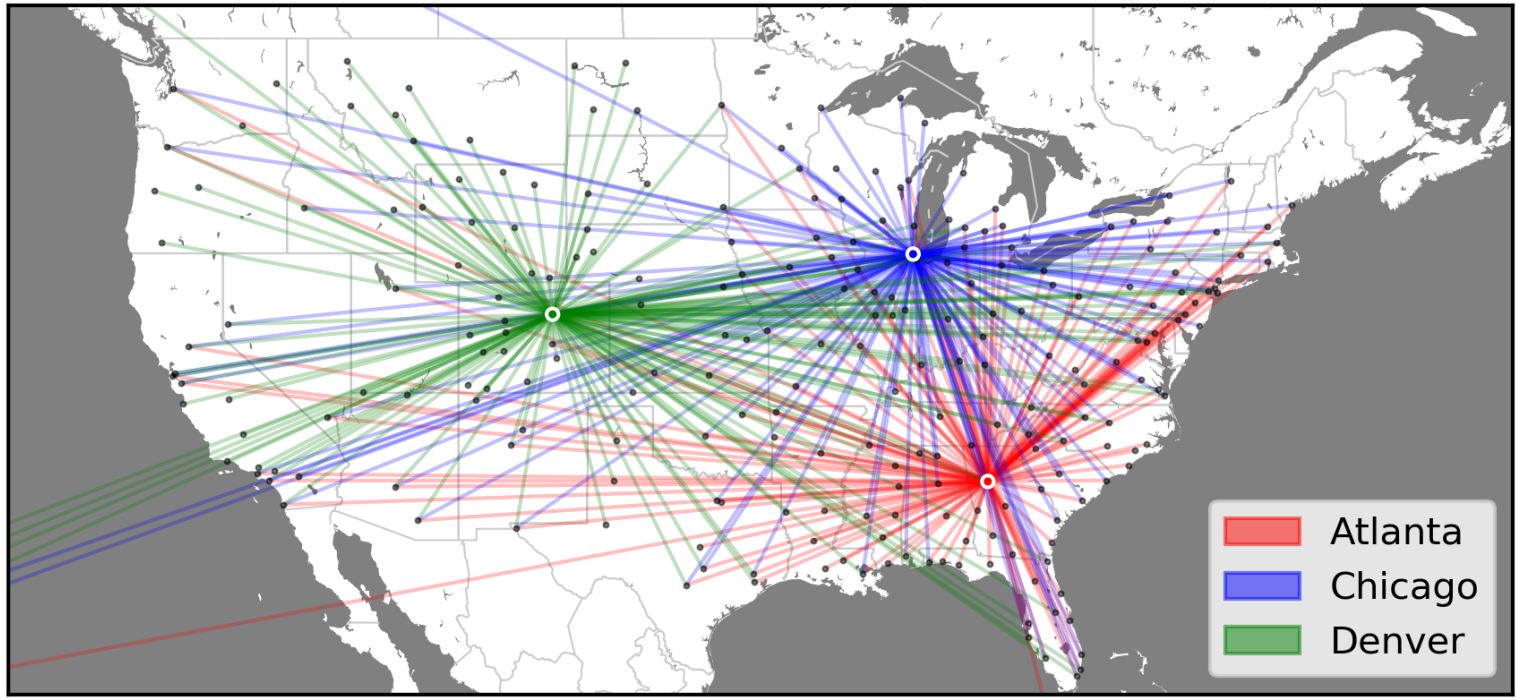
Analisi e Visualizzazione delle Reti Complesse

NS06 - Hubs, Closeness, Betweenness, Robustness

Prof. Rossano Schifanella



Real networks are heterogeneous



Some nodes (and links) are much more important (**central**) than others!

UNIVERSITÀ DI TORINO

Centrality measures

- **Centrality**: it measures the **importance** of a node
- It can be quantified by different **measures**:
  - **Degree**
  - **Closeness**
  - **Betweenness**

Degree

- The **degree of a node** k_i is the **number of neighbors** of the node i
- High-degree nodes are called **hubs**
- **Average degree** of an undirected network:

$$\langle k \rangle = \frac{\sum_i k_i}{N} = \frac{2L}{N}$$

In NetworkX:

```
G.deegree(2) # returns the degree of node 2

G.degree()   # returns a dict with the degree of all nodes of G
```

Closeness

Idea: **A node is more central the closer it is to the other nodes, on average**

$$g_i = \frac{1}{\sum_{j \neq i} l_{ij}}$$

where l_{ij} is the distance between nodes i and j

In a normalized form (discounting the graph size):

$$g_i = \frac{N-1}{\sum_{j \neq i} l_{ij}} = \frac{1}{\frac{\sum_{j \neq i} l_{ij}}{N-1}}$$

That is the inverse of the average distance from the focal node i to the rest of the network

```
nx.closeness_centrality(G, node)   # returns the closeness centrality of node i
```

Betweenness

Idea: **A node is more central the more often it is crossed by shortest paths**

$$b_i = \sum_{h \neq j \neq i} \frac{\sigma_{hj}(i)}{\sigma_{hj}}$$

$$\sigma_{hj} = \text{number of shortest paths from } h \text{ to } j$$

$$\sigma_{hj}(i) = \text{number of shortest paths from } h \text{ to } j \text{ running through } i$$

This measure **depends on the size of the network**.

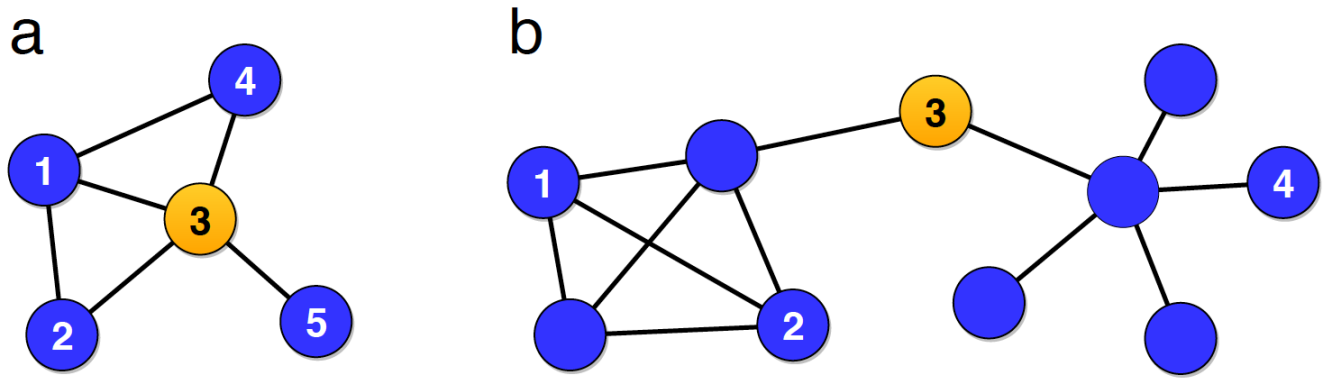
We can normalize the formula by the **maximum number of paths that could go through node** i, e.g., the number of pairs of nodes excluding i:

$$\frac{b_i}{\binom{N-1}{2}} = \sum_{h \neq j \neq i} \frac{2\sigma_{hj}(i)}{(N-1)(N-2)\sigma_{hj}}$$

Betweenness

Hubs usually have a **high betweenness** (a).

There can be **nodes with high betweenness that are not hubs** (b).



Edge Betweenness

Betweenness can be easily extended to edges.

Idea: **The fraction of shortest paths among all possible node pairs that pass through the link**

```
nx.betweenness_centrality(G)       # dict nodes: betweenness centrality

nx.edge_betweenness_centrality(G)  # dict links: betweenness centrality
```

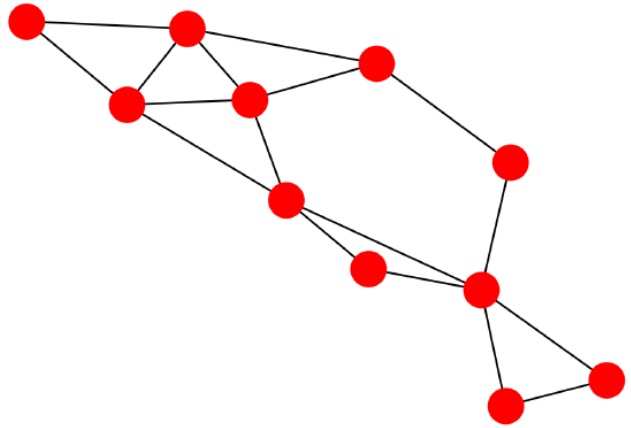
Centrality distributions

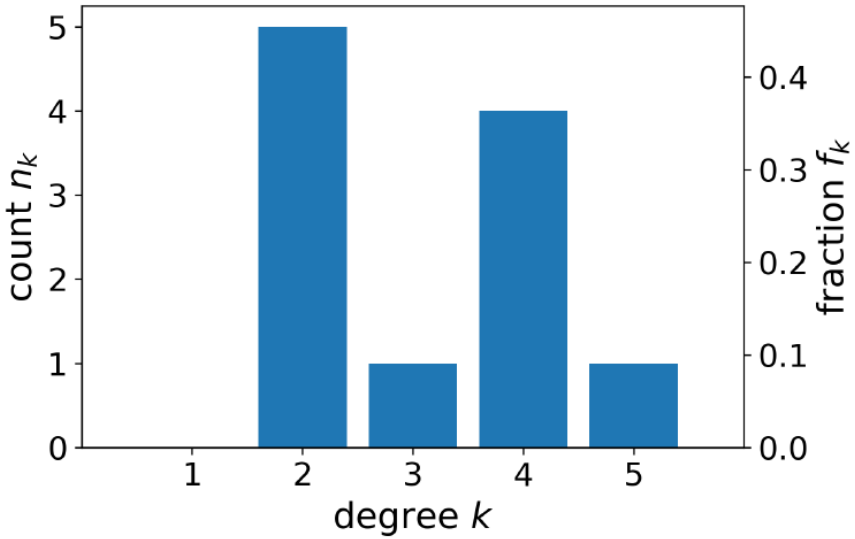
On small networks, it makes sense to ask which nodes or links are most important.

On large networks, it does not.

Solution: **A statistical approach**

> Instead of focusing on individual nodes and links, we consider classes of nodes and links with similar properties.





$n_k =$ number of nodes with degree k

$f_k = \frac{n_k}{N} =$ frequency of degree k

For large N, i.e., $\lim_{N\to\infty}$, the frequency f_k converges to the probability p_k of an observartion having degree k

Probability distribution:
Plot of probability p_k versus k

When the quantity under exam is not an integer, e.g., betweenness centrality, we can divide the range of values into **disjoint intervals**, or **bins**.

We can similarly count the number of observations falling within each bin

We can use this technique whenever we are interested in **ranges of values**, even if the values are integers.

Complementary Cumulative Distribution Function (CCDF)

- **Complementary Cumulative Distribution Function** $P(x)$**:**
  probability that the variable takes values larger than x as a function of x
- **How to compute it:**
  by summing the frequencies of the variable inside the intervals to the right of x

$$P(x) = \sum_{v>x} f_v$$

- The CCDF is the complement of the Cumulative Distribution Function, i.e., in a not-too-formal notation $CCDF(x) = P(X > x) = 1 - CDF(x) = 1 - P(X \leq x)$
- The CCDF is particularly useful in contexts where the focus is on the tail of the distribution, averaging out the noise due to rare high-value events.

Logarithmic scale

Question:

How do you plot a probability distribution if the variable spans a large range of values, from small to (very) large?

Answer:

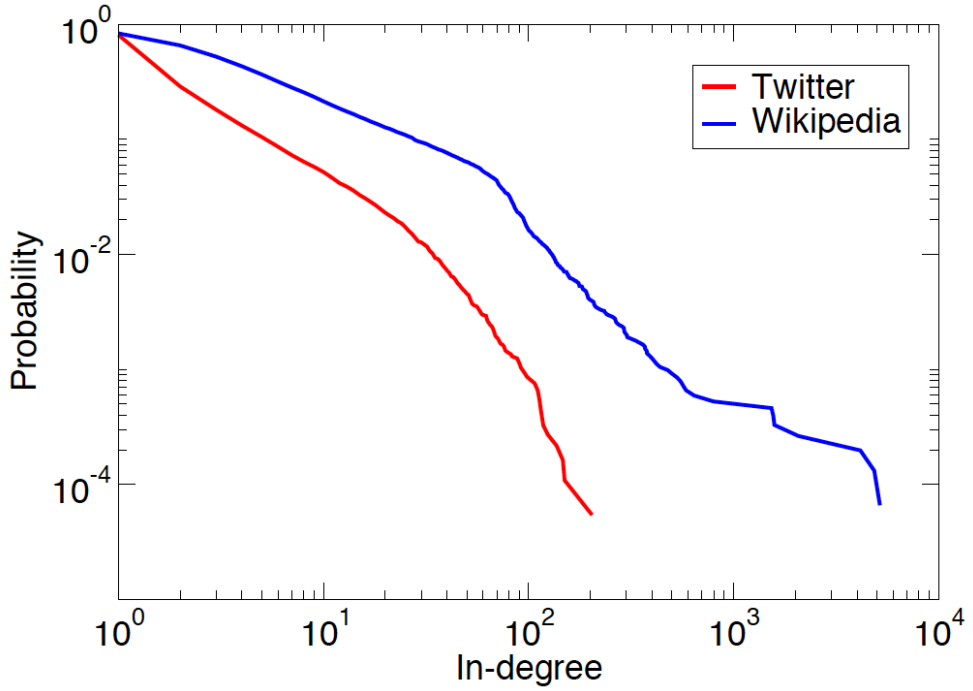
Use a **logarithmic scale**

How to do it:

Report the logarithms of the values on the x- and y-axes.

$$\log_{10} 10 = 1$$
$$\log_{10} 1,000 = \log_{10} 10^3 = 3$$
$$\log_{10} 1,000,000 = \log_{10} 10^6 = 6$$

In-degree distributions



Heavy-tail distributions: the variable goes from small to large values

Heterogeneity Parameter κ

The **heterogeneity parameter** measures the **breadth** of the degree distribution, comparing the variability of the degree across nodes to the average degree

Let us define the **average squared degree** $\langle k^2 \rangle$ as:

$$\langle k^2 \rangle = \frac{k_1^2 + k_2^2 + \cdots + k_{N-1}^2 + k_N^2}{N} = \frac{\sum_i k_i^2}{N}$$

$$\langle k \rangle = \frac{\sum_i k_i}{N} = \frac{2L}{N}$$

$$\kappa = \frac{\langle k^2 \rangle}{\langle k \rangle^2}$$

If most degrees have the same value, say k_0:

$$\langle k \rangle \approx k_0, \ \langle k^2 \rangle \approx k_0^2 \implies \kappa \approx 1$$

If the distribution is very heterogeneous:

$$\kappa \gg 1$$

If a network is directed, we have to consider two distributions, the **in-degree** and **out-degree** distributions, defined as the probability that a randomly chosen vertex has a given in- or out-degree, respectively.

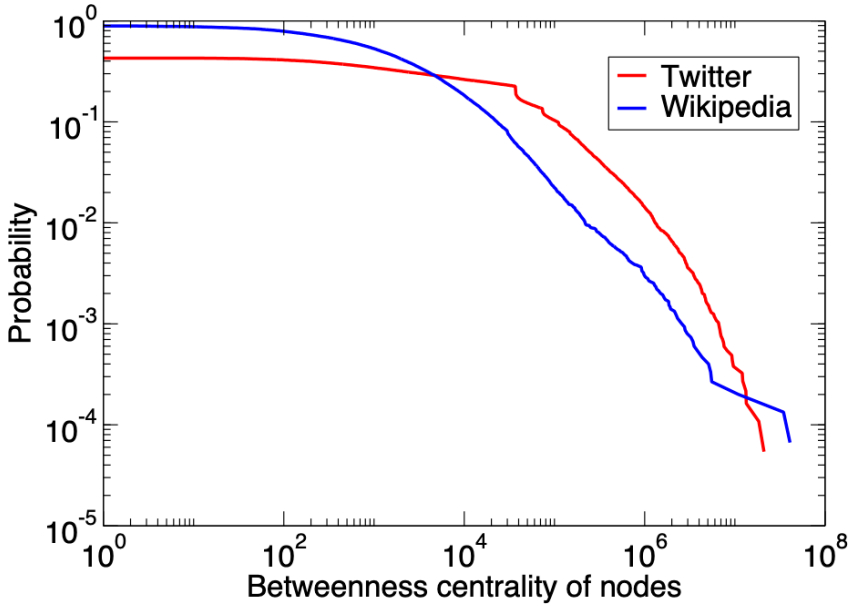
UNIVERSITÀ DI TORINO

Degree centrality

Network	Nodes (N)	Links (L)	Average degree ($\langle k \rangle$)	Maximum degree (k_{max})	Heterogeneity parameter (κ)
Facebook Northwestern Univ.	10,567	488,337	92.4	2,105	1.8
IMDB movies and stars	563,443	921,160	3.3	800	5.4
IMDB co-stars	252,999	1,015,187	8.0	456	4.6
Twitter US politics	18,470	48,365	2.6	204	8.3
Enron Email	36,692	367,662	10.0	1,383	14.0
Wikipedia math	15,220	194,103	12.8	5,171	38.2
Internet routers	190,914	607,610	6.4	1,071	6.0
US air transportation	546	2,781	10.2	153	5.3
World air transportation	3,179	18,617	11.7	246	5.5
Yeast protein interactions	1,870	2,277	2.4	56	2.7
C. elegans brain	297	2,345	7.9	134	2.7
Everglades ecological food web	69	916	13.3	63	2.2

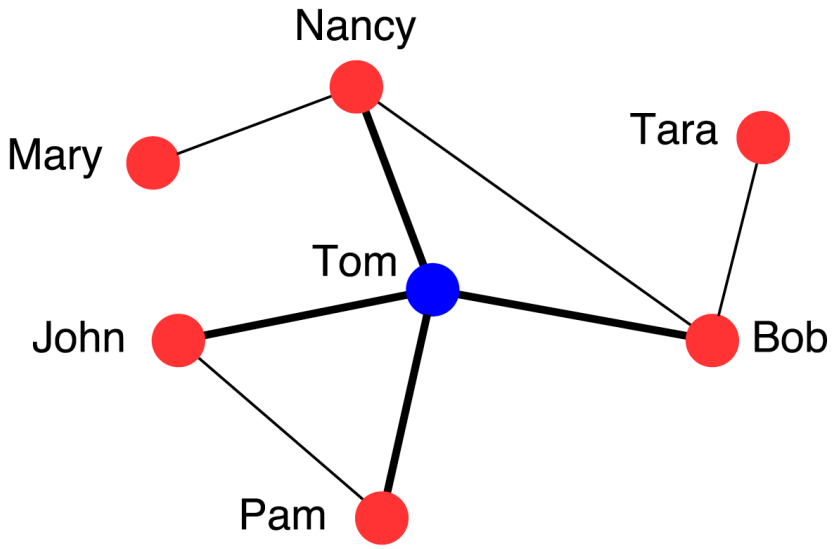
Betweenness distributions

- One can analyze the distributions of other properties besides the degree, e.g., betweenness
- It turns out that the **degree is usually correlated with other centrality measures**



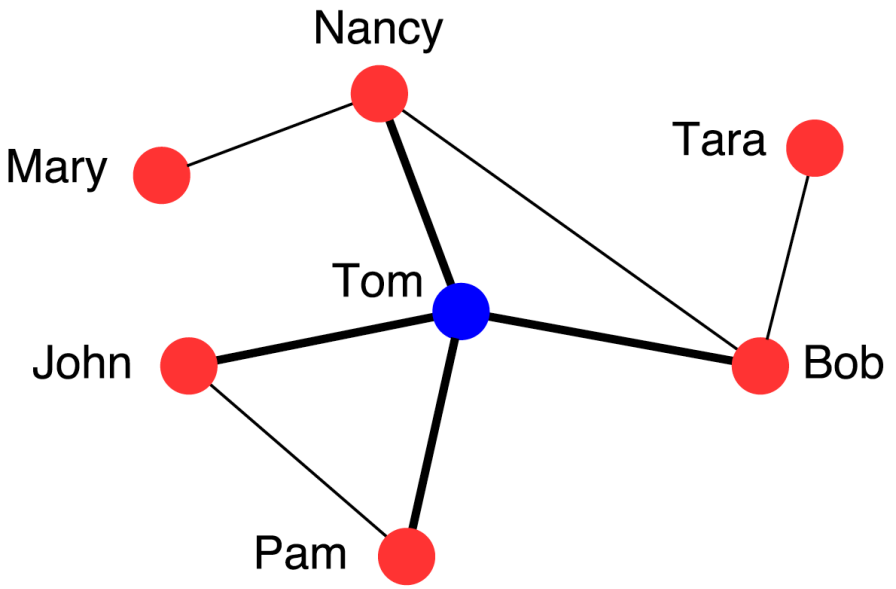
- **Heavy-tail distributions:** the variable goes from small to large values

Friendship paradox



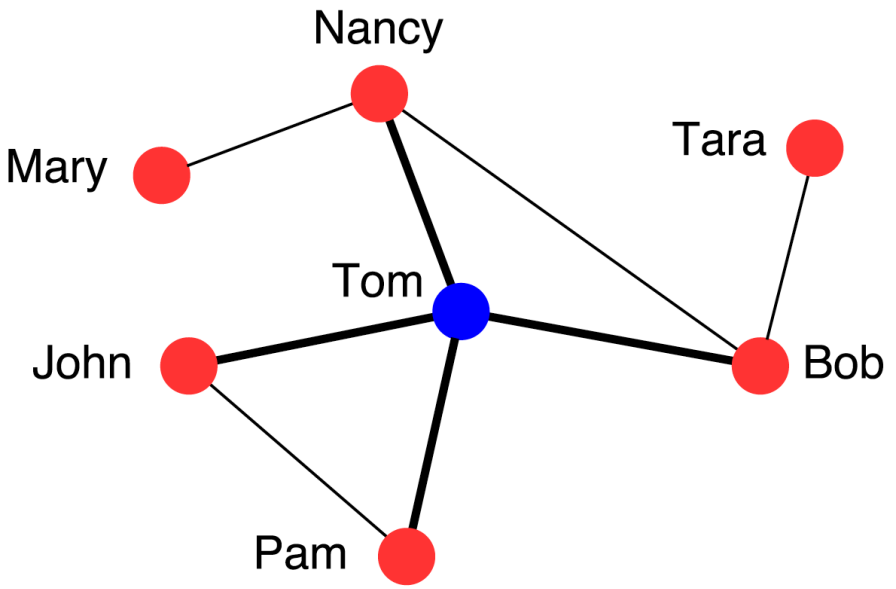
- By choosing a node at random, Tom has **the same chance** to be picked as everybody else
- By choosing at random a friend of a random individual (same as choosing a link at random), Tom has **a higher chance** of being picked than everybody else

Friendship paradox



By following links, **the chance to hit a hub increases.**

Friendship paradox



The average degree of a node = **2.29**

The average degree of the neighbors of a node = **2.83 > 2.29**

Our friends have more friends than we do, on average **(friendship paradox)**

Friendship paradox

Question:

Where does the friendship paradox come from?

Answer:

1. **By averaging the degree of the nodes, we pick them at random**
2. **By averaging the degree of the neighbors, we choose them by following links**: nodes with degree k will be counted k times, which inflates the average

- In other words, the Friendship Paradox is thus due to **sampling**. The two averages are computed by sampling the node degrees differently:
  - **uniformly** for average degree,
  - **proportionally to the degree** for the neighbors' average degree.

The more hubs, the stronger the effect.

Simple proof

$$\mu_f = \frac{\sum k_i}{N}$$
$$\mu_{fof} = \frac{\sum k_i^2}{\sum k_i}$$

From the definition of variance

$$\delta^2 = \frac{\sum k_i^2}{N} - \mu^2 \Rightarrow \sum k_i^2 = (\mu^2 + \delta^2)N$$

Then

$$\mu_{fof} = \frac{\sum k_i^2}{\sum k_i} = \frac{(\mu^2 + \delta^2)N}{\mu_f N} = \mu_f + \frac{\delta^2}{\mu_f}$$

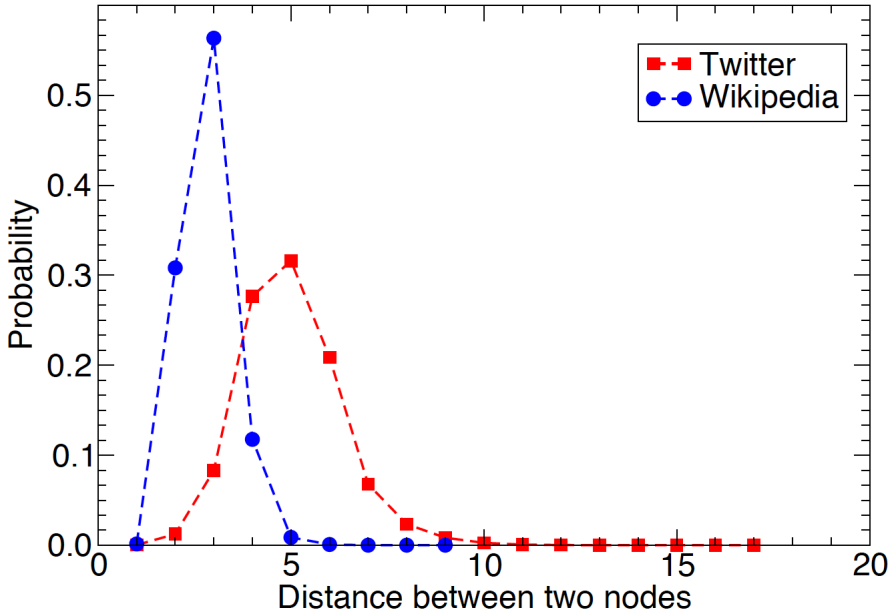
Ultra-small worlds

In real networks, many of the shortest paths go through hubs.

- Example: air transportation
- There may be no routes between airports A and B (if they are small), but it may be possible to go from A to B via a hub airport C

The small-world property is typical of most networks of interest

With **hubs**, **paths are ultra-short** (ultra-small world)



Robustness

A system is **robust** if the **failure of some of its components does not affect its function**.

Question:

How can we define the robustness of a network?

Answer:

We remove nodes and/or links and see what happens to its structure

Key point:

Connectedness

If the Internet were not connected, transmitting signals (e.g., emails) between routers in different components would be impossible.

Robustness

Robustness test:

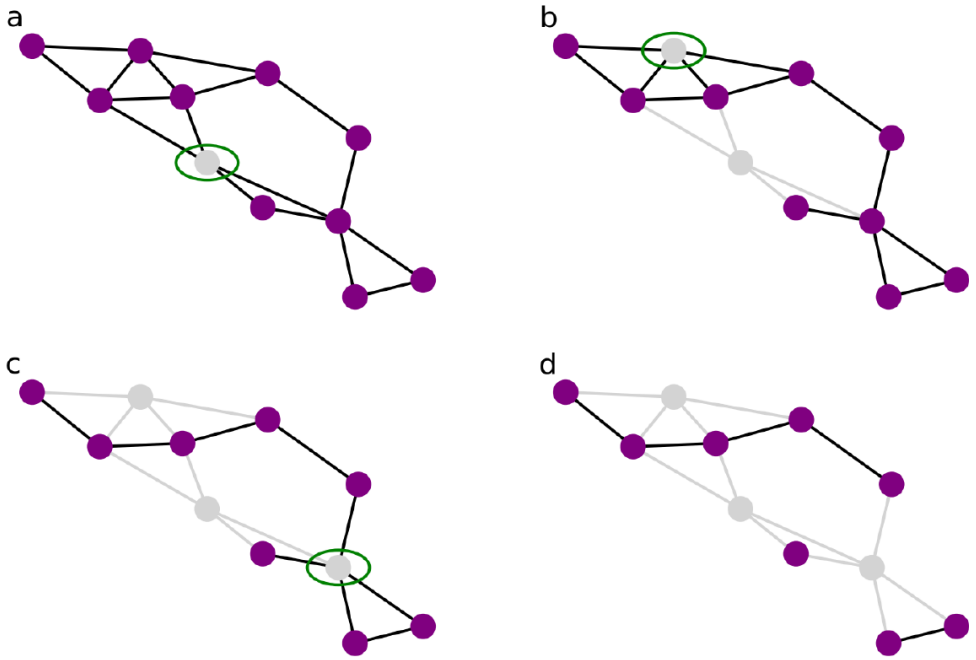
Checking how the connectedness of the network is affected as nodes are removed.

How to do it:

Plot the relative size S of the largest connected component as a function of the fraction of removed nodes.

- We suppose that the network is initially connected
  - there is only one component, and $S = 1$
- As more and more nodes (and their links) are removed, the network is progressively broken up into components, and S goes down.

Robustness



Robustness

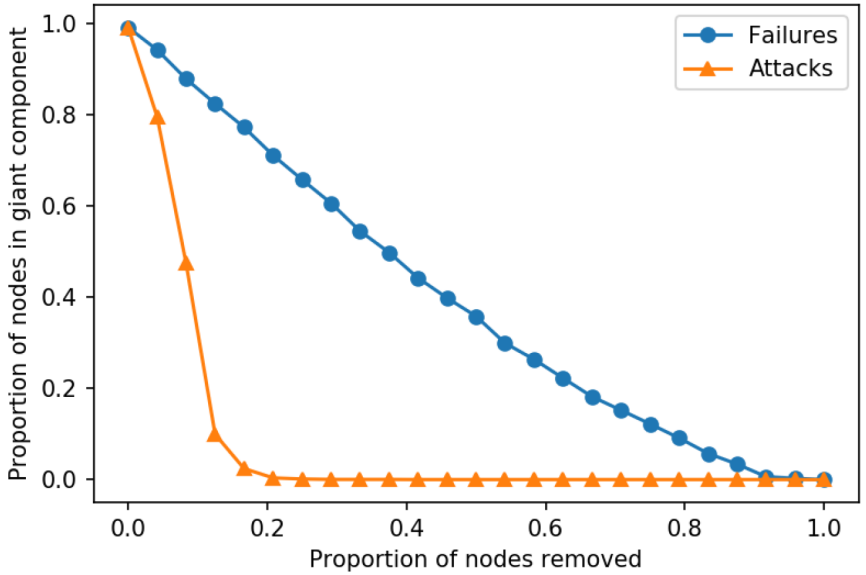
Two strategies:

1. **Random failures:** nodes break down randomly, so they are all chosen with the **same probability**
2. **Attacks:** hubs are deliberately targeted — the **larger** the **degree**, the higher the probability of removing the node

In the first approach: we remove a fraction f of **randomly** chosen nodes.

In the second approach: we remove the fraction f of **nodes with the largest degree**.

Robustness



Real networks are:

1. **Robust against random failures**
2. **Fragile against targeted attacks**

Core decomposition

Core:

Dense part of the network, with high degree nodes

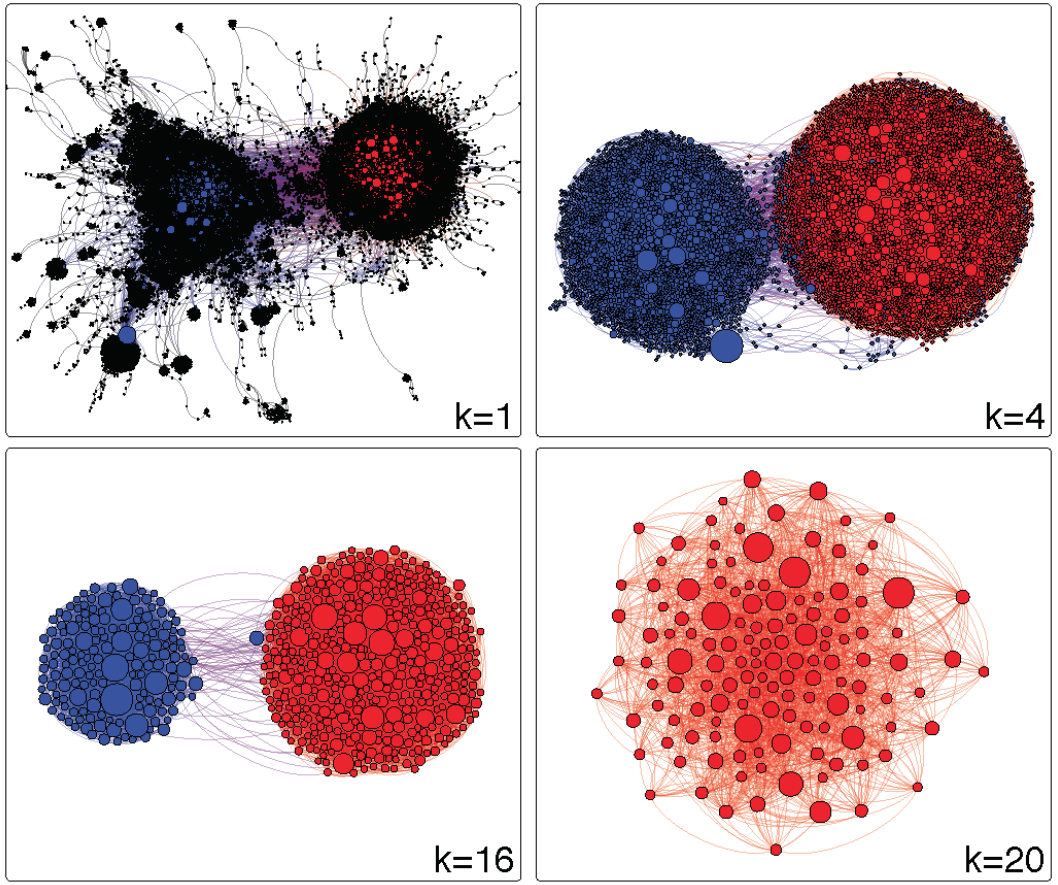
Core decomposition:

Procedure to **identify denser and denser cores by removing nodes of progressively higher degree**. If we remove all nodes with degree $(k{-}1)$ or lower, the remaining portion of the network is called **k-core**

K-core decomposition procedure: (start with $k = 0$)

1. Recursively remove all nodes with degree k, until none are left
2. The set of removed nodes is the **k-th shell**, the remaining ones form the **(k+1)-core**
3. If there is no node left, terminate. Otherwise, increment k by one and repeat from step 1

Core decomposition



Core decomposition

Core decomposition helps to visualize large networks, by pruning low-degree nodes and showing only the densest parts

In NetworkX:

```
nx.core_number(G)   # return dict with core number of each node
nx.k_shell(G,k)     # subnetwork induced by nodes in k-shell
nx.k_core(G,k)      # subnetwork induced by nodes in k-core
nx.k_core(G)        # innermost (max-degree) core subnetwork
```

UNIVERSITÀ DI TORINO

Calculating betweenness

Computing edges' betweenness

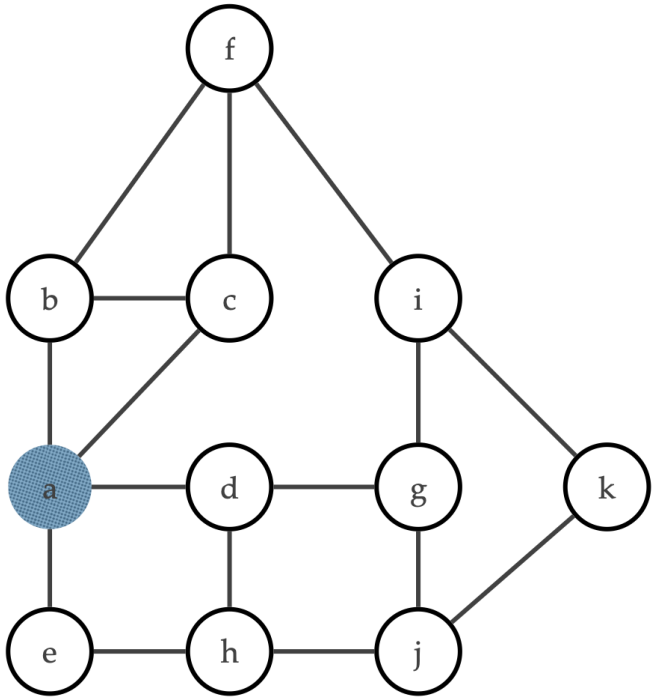
For each node $i \in N$

1. perform BFS starting from i
2. count the number of shortest paths from i to all the other nodes
3. calculate the amount of flow from i to all other nodes that use each edge

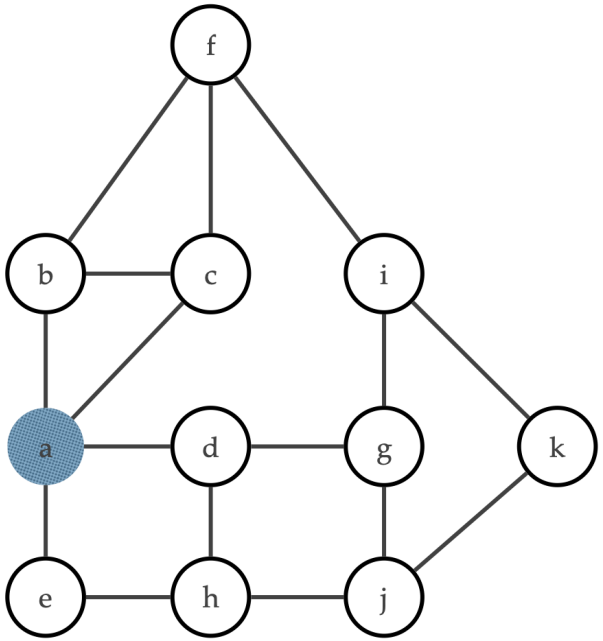
Reference:

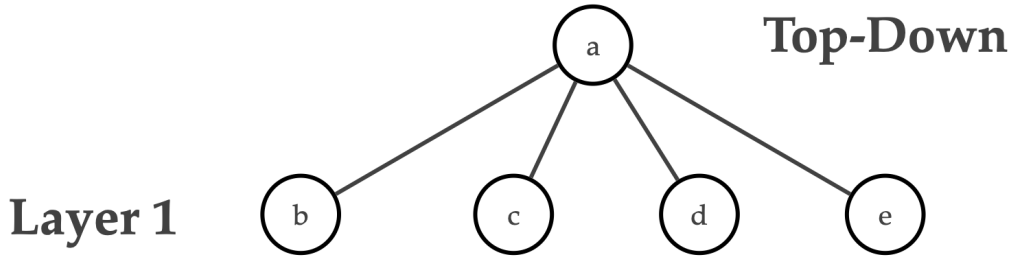
- Ulrik Brandes (2001) A faster algorithm for betweenness centrality, The Journal of Mathematical Sociology, 25:2, 163-177, DOI: 10.1080/0022250X.2001.9990249

Step 1: BFS from node a

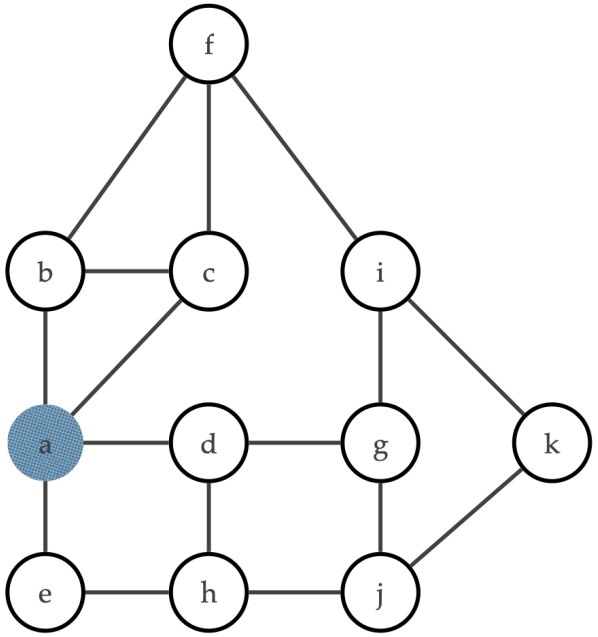


Step 1: BFS from node a



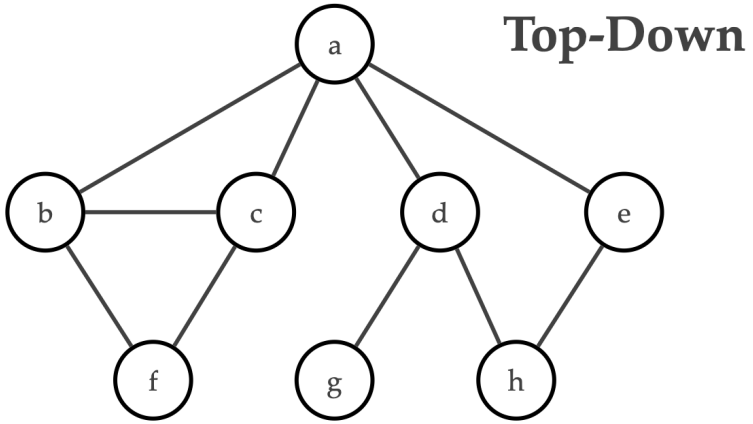


Step 1: BFS from node a

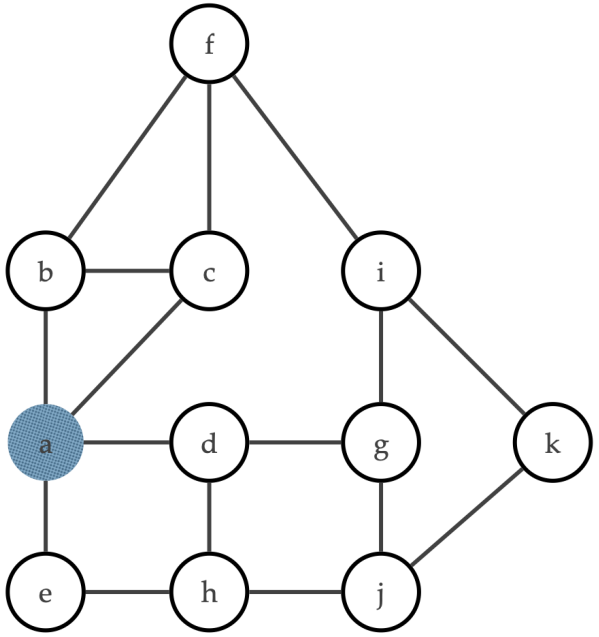


Layer 1

Layer 2



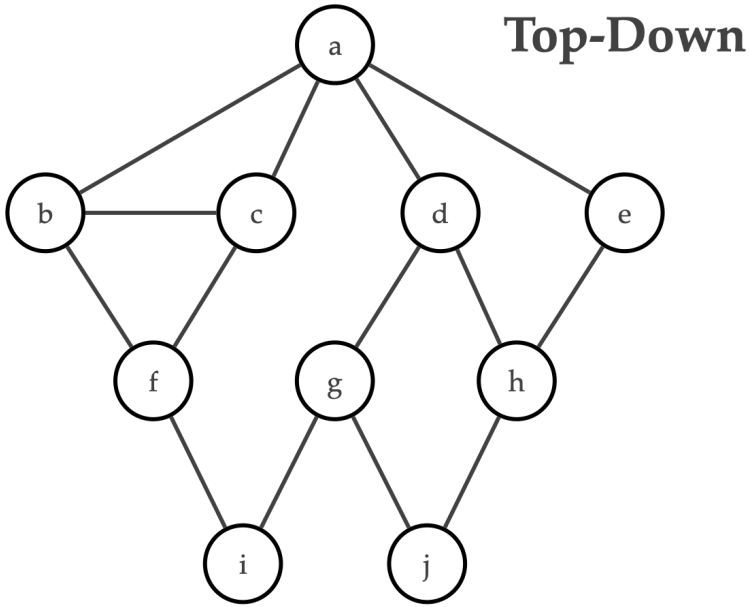
Step 1: BFS from node a



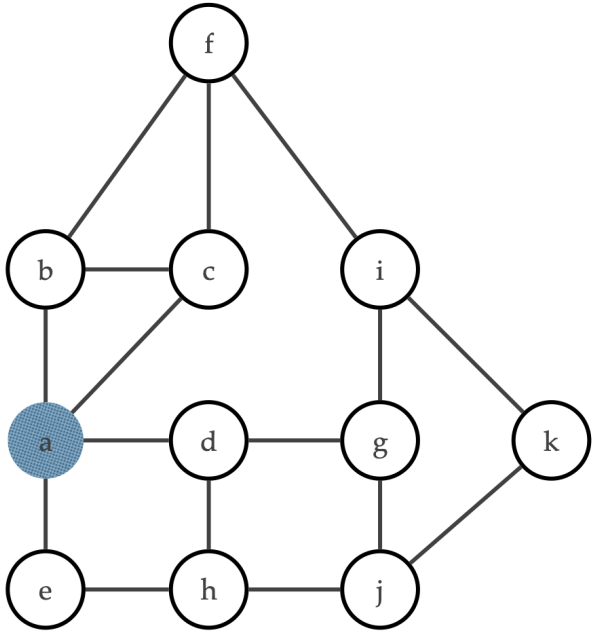
Layer 1

Layer 2

Layer 3



Step 1: BFS from node a

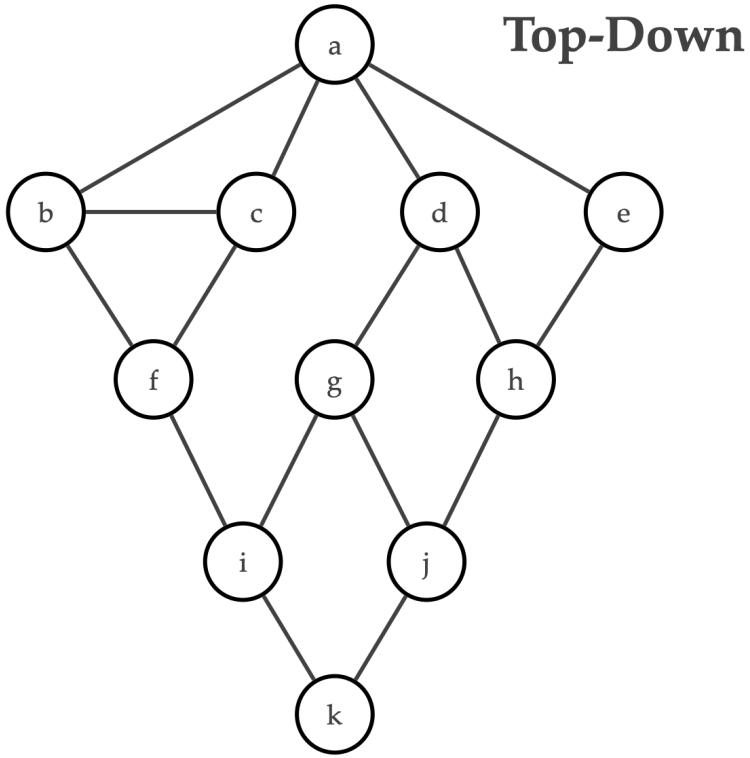


Layer 1

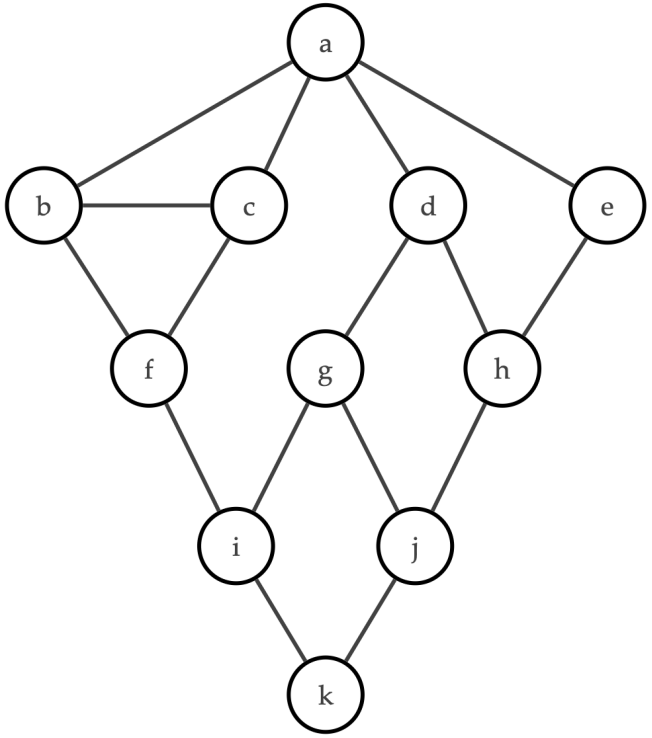
Layer 2

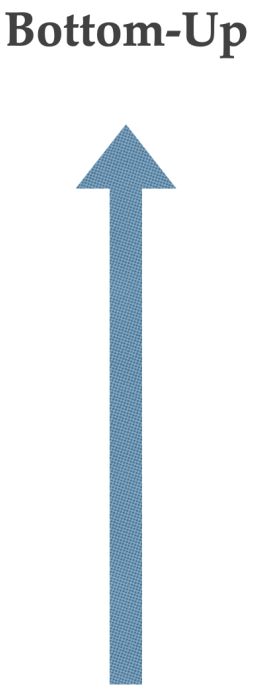
Layer 3

Layer 4

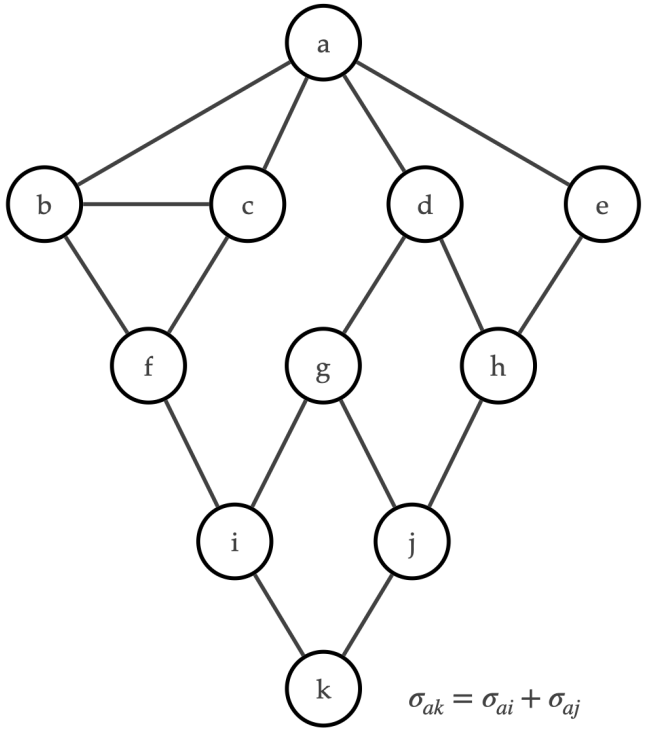


Step 2: count the # shortest paths from a



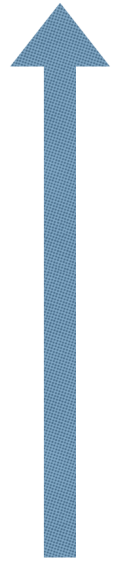


Step 2: count the # shortest paths from a

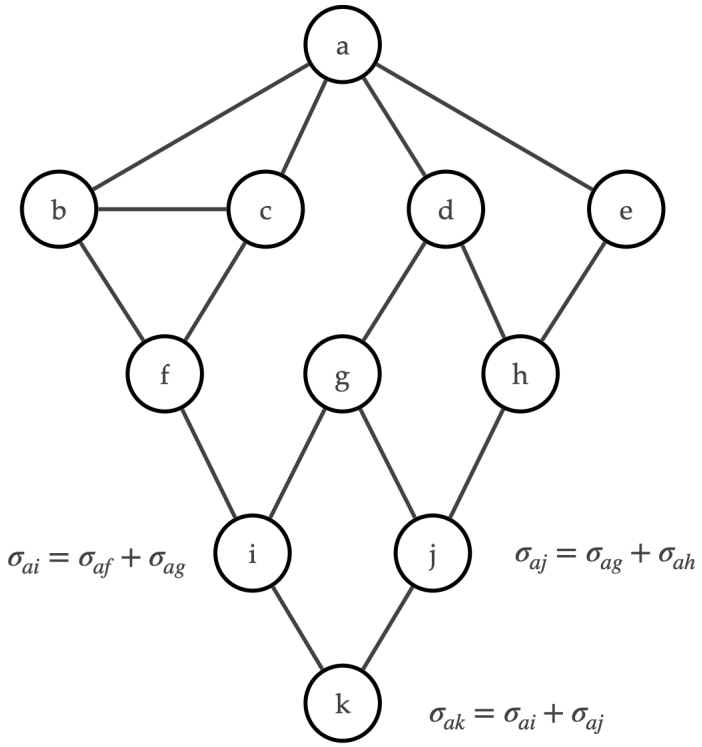


$\sigma_{ak} = \sigma_{ai} + \sigma_{aj}$

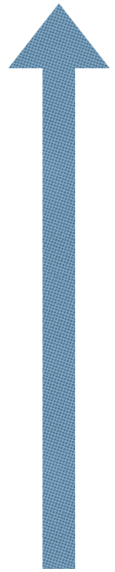
Bottom-Up



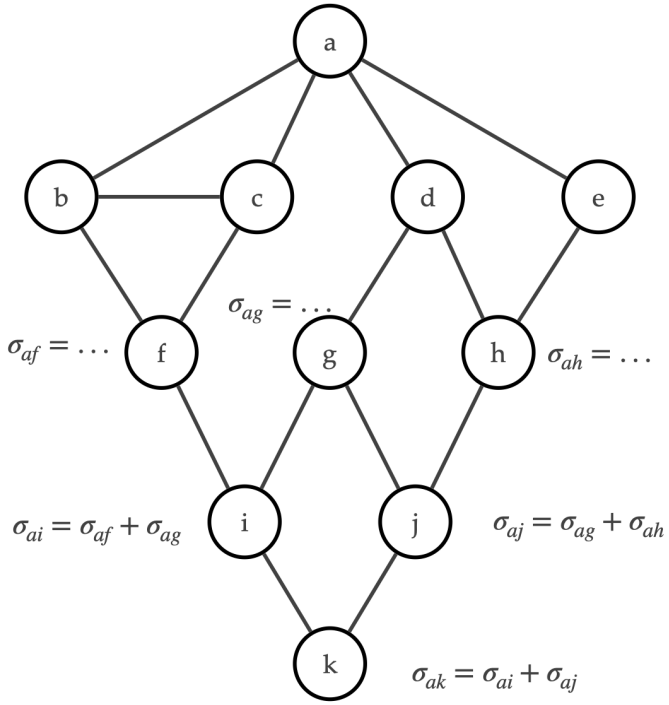
Step 2: count the # shortest paths from a

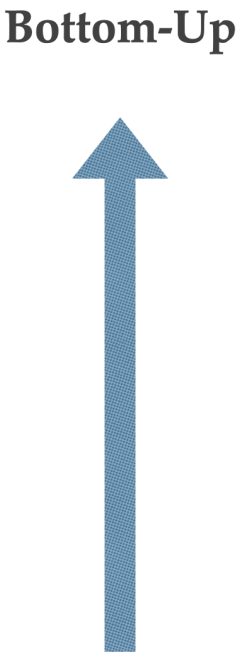


Bottom-Up

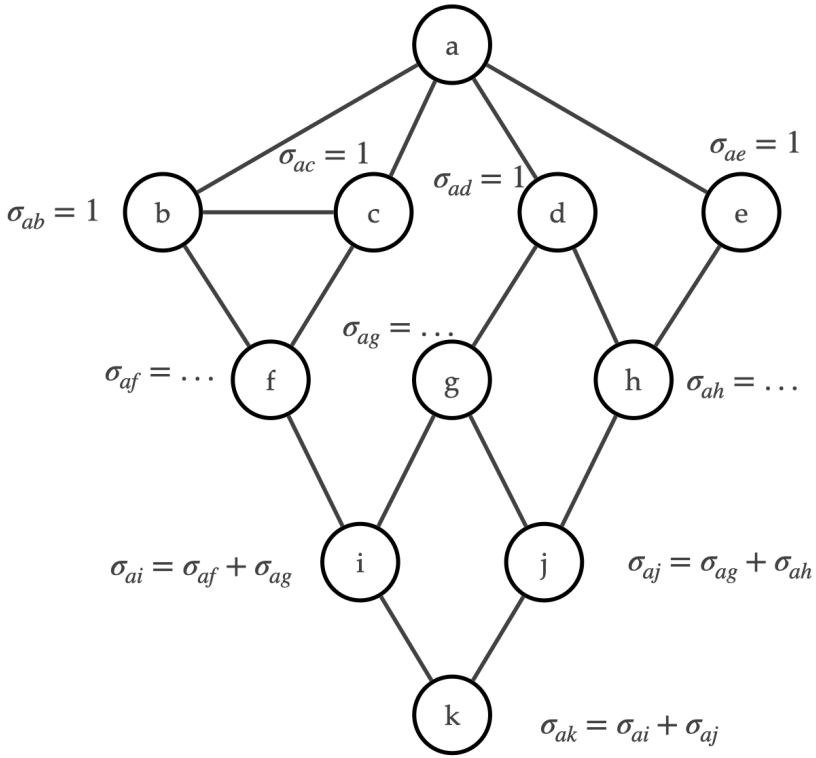


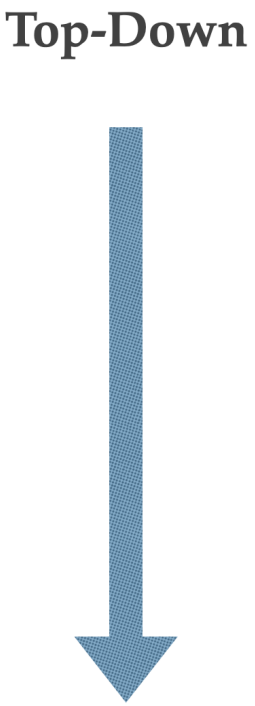
Step 2: count the # shortest paths from a



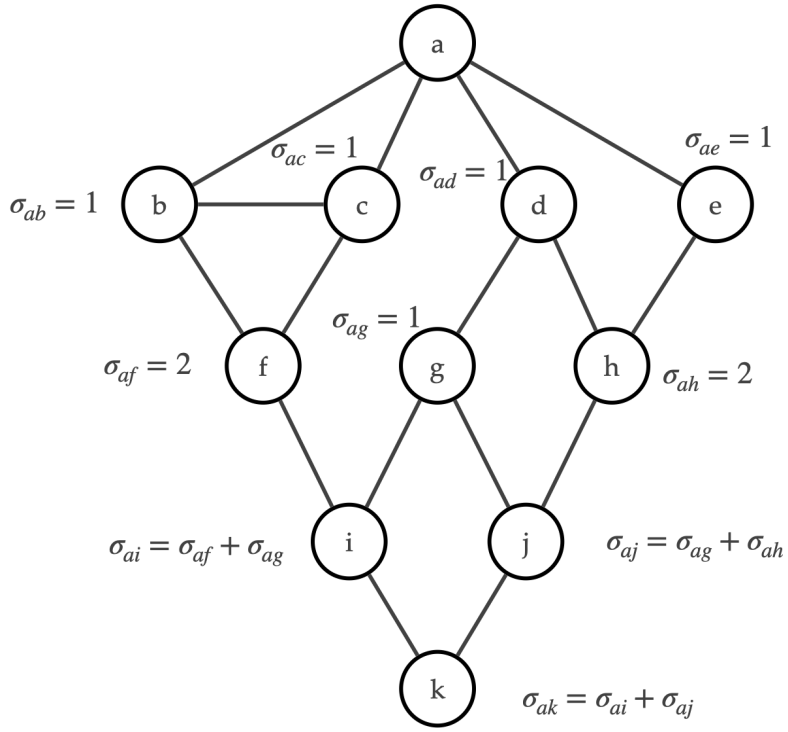


Step 2: count the # shortest paths from a

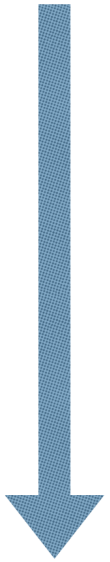




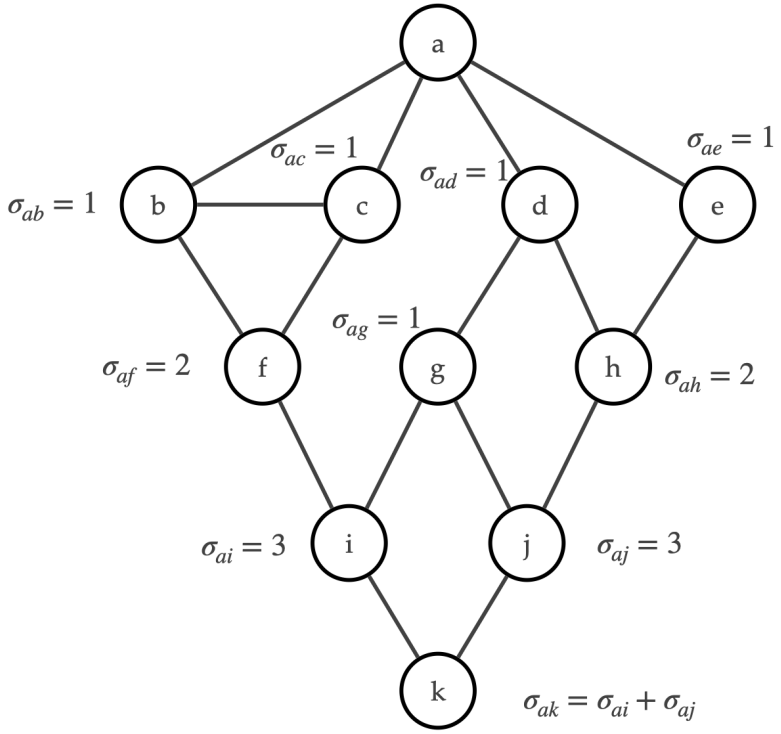
Step 2: count the # shortest paths from a

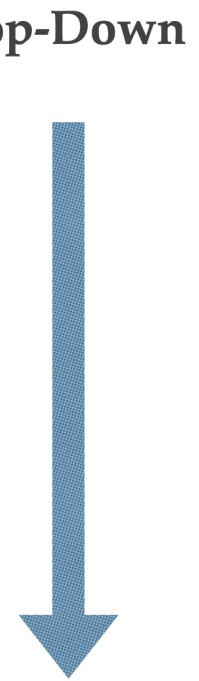


Top-Down

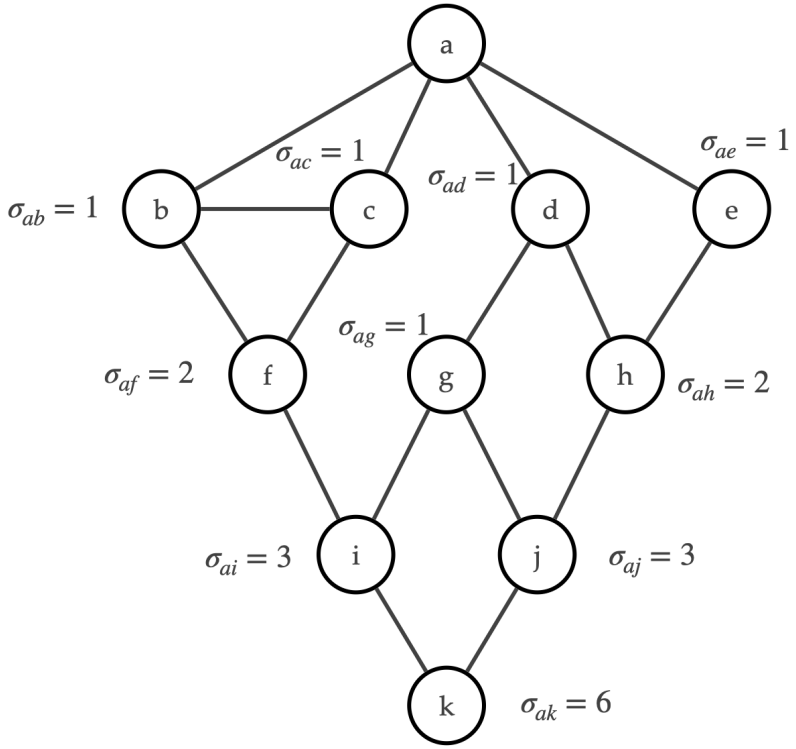


Step 2: count the # shortest paths from a

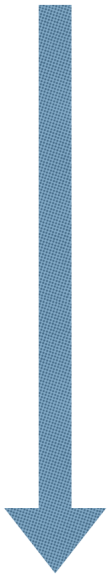




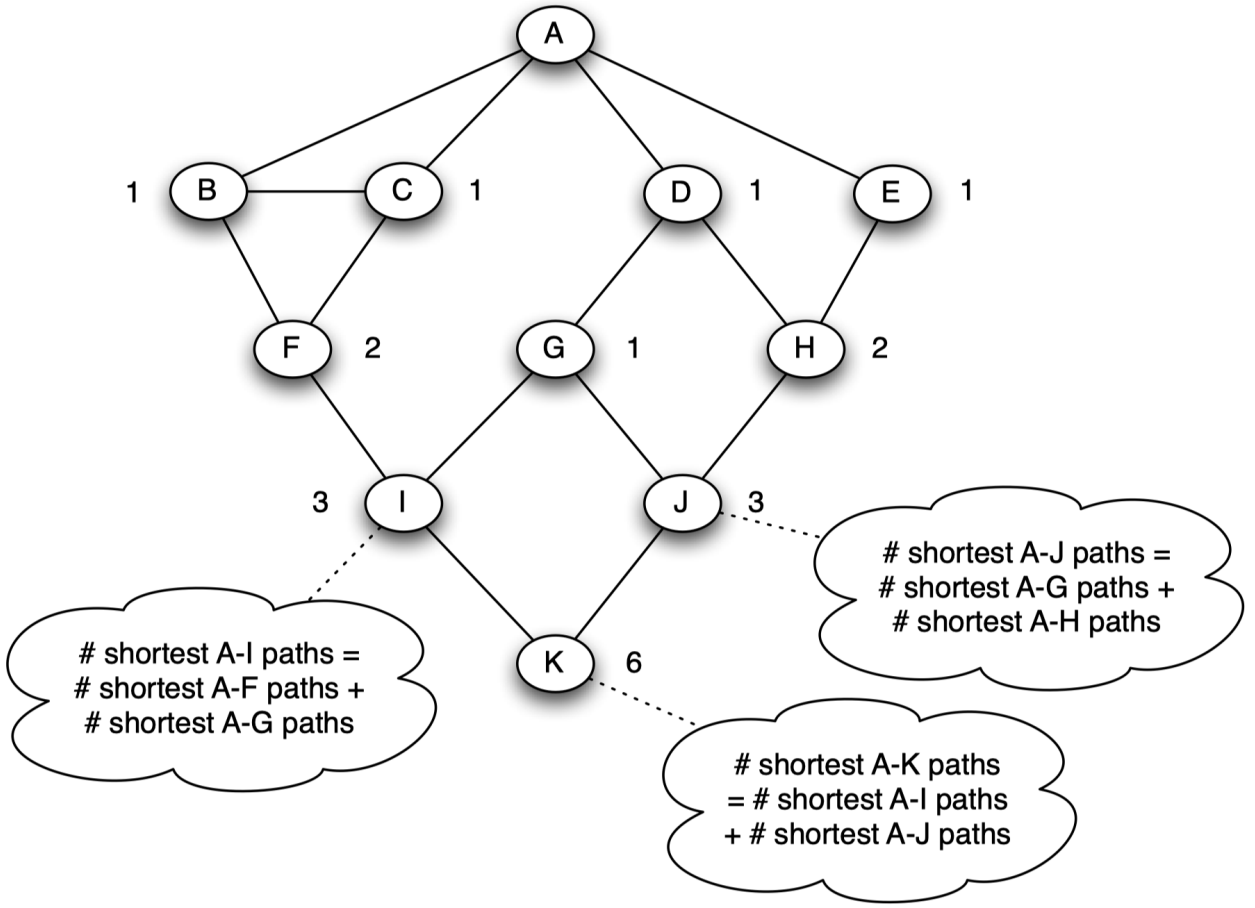
Step 2: count the # shortest paths from a



Top-Down



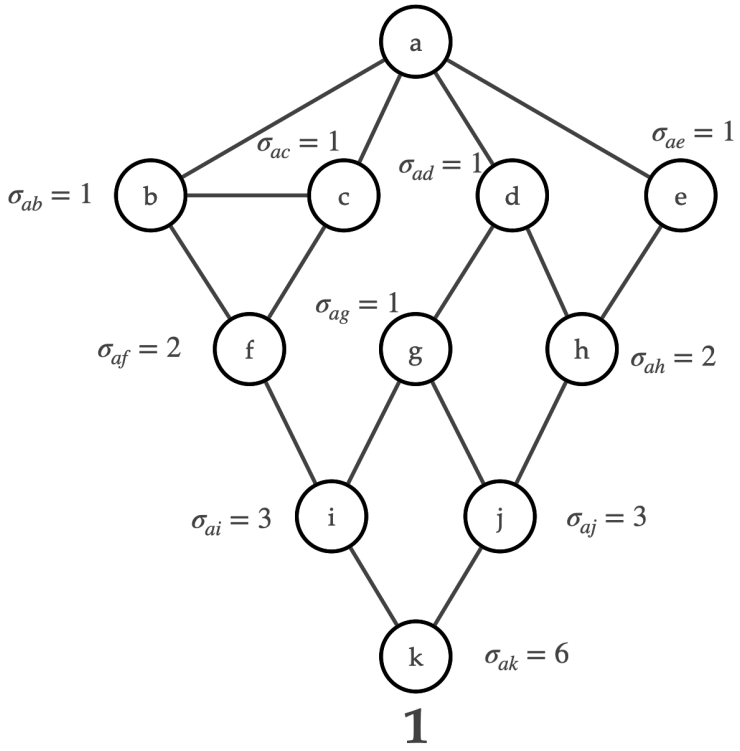
Step 2: Overview



Step 3: computing the flow

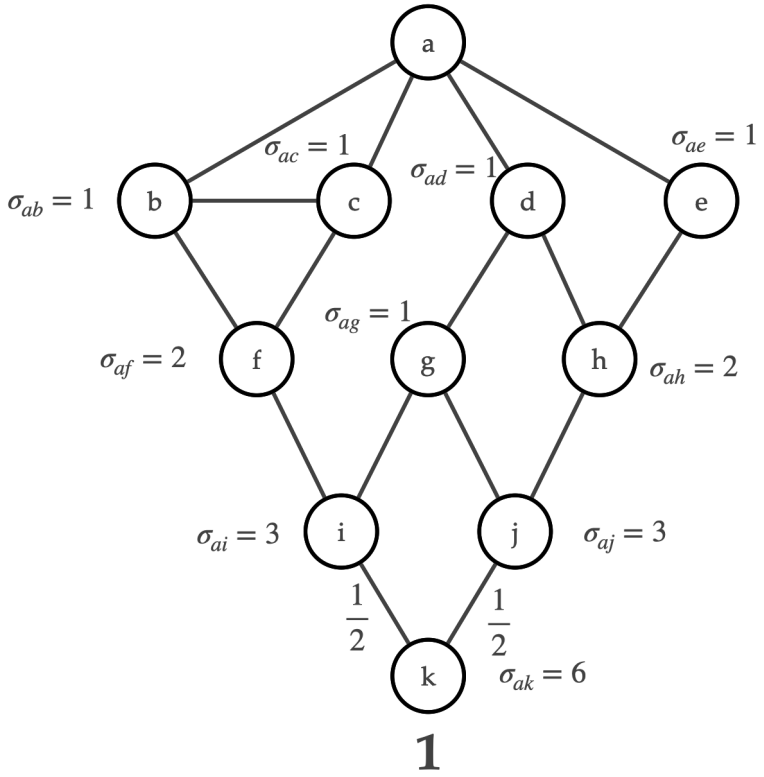
For each node i in the BFS structure, we add up all the flow arriving from edges directly below i, plus 1 for the flow destined for i itself.

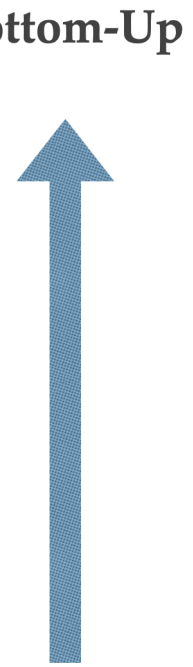
We then divide this up over the edges leading upward from i, in proportion to the number of shortest paths coming through each.



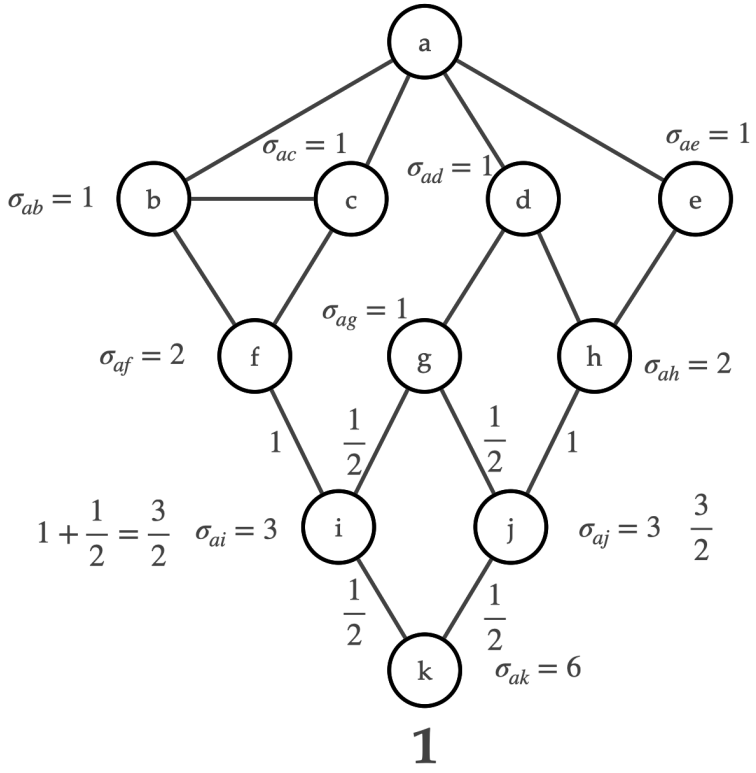
Bottom-Up

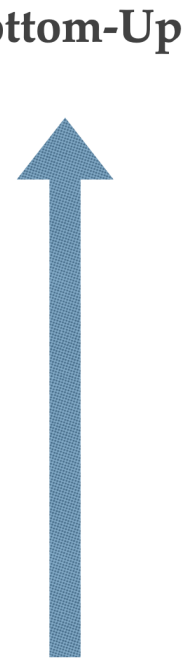
Step 3: computing the flow



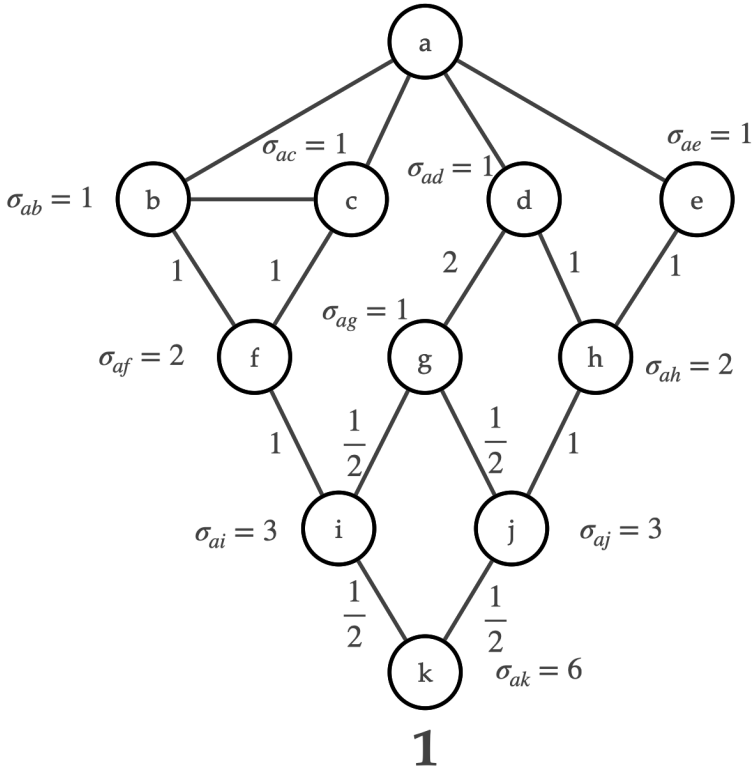


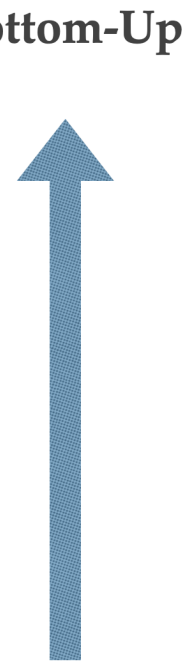
Step 3: computing the flow



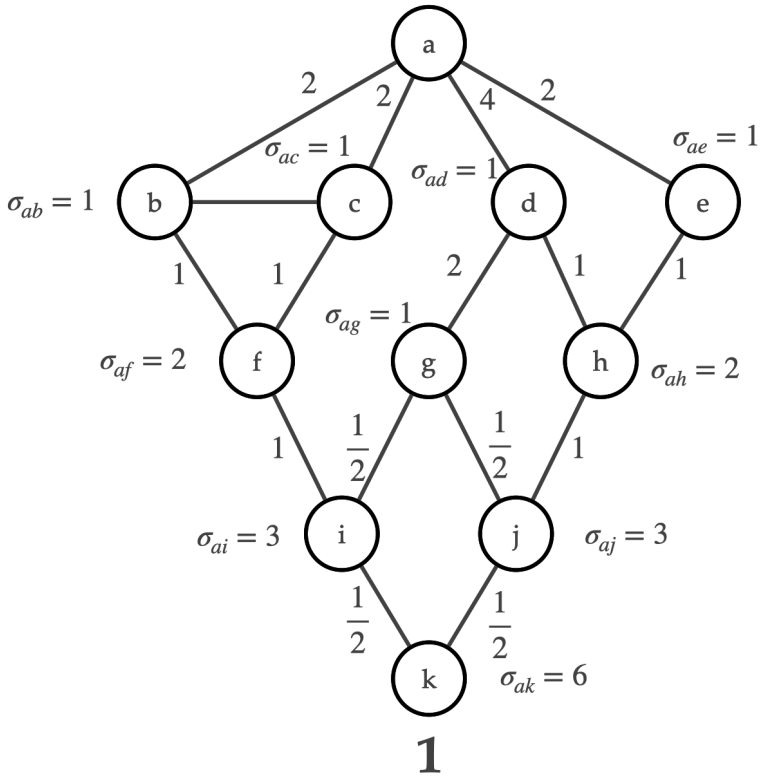


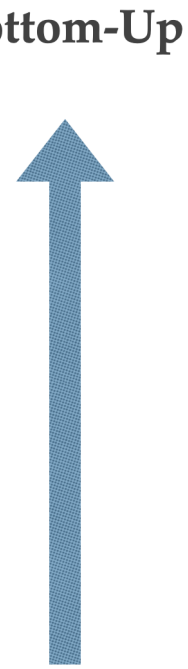
Step 3: computing the flow



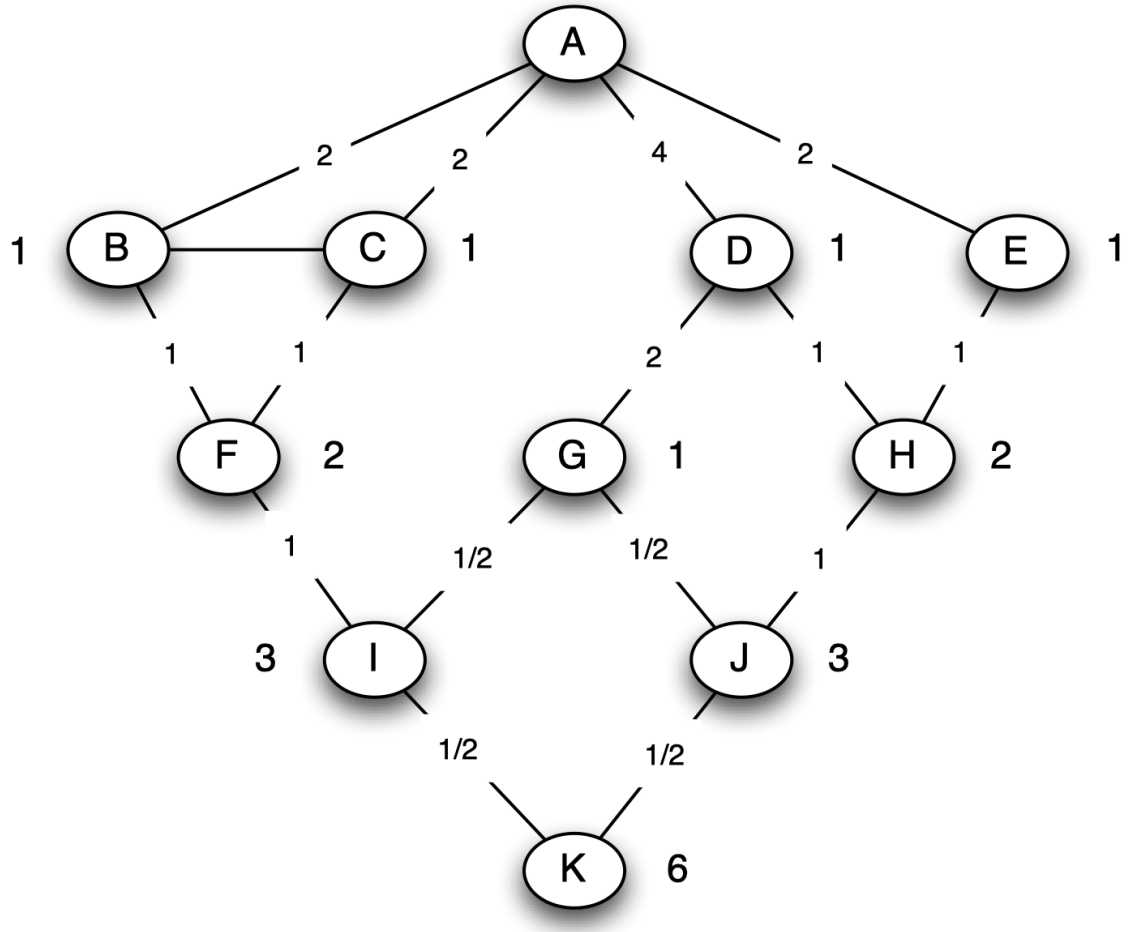


Step 3: computing the flow





Step 3: Overview



UNIVERSITÀ DI TORINO

Output: betweenness for each edge

- Repeat for all the nodes in G
- For each edge (i,j):= sum up all the 'flow' values / 2
- Complexity of Girvan-Newman:
  - $O(N^3)$ for dense graphs
  - $O(N^2)$ for sparse graphs
- We can do better:
  - Leuven Algorithm: O(N)

Additional readings

- Linton C. Freeman, Centrality in social networks. A conceptual clarification, Social networks, vol. 1, no 3, 1979, p. 215–239
- Ulrik Brandes (2001) A faster algorithm for betweenness centrality, The Journal of Mathematical Sociology, 25:2, 163-177
- Newman, Mark EJ, and Michelle Girvan. "Finding and evaluating community structure in networks." Physical review E 69.2 (2004): 026113

UNIVERSITÀ DI TORINO

Reading material

References

[ns1] **Chapter 3**

[ns2] **Chapter 3.6B: Advanced Material: Computing Betweenness Values**

Q&A