# Biological Networks Analysis

## Dijkstra's algorithm and Degree Distribution

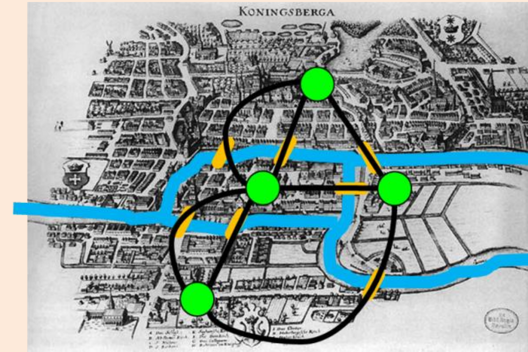
Genome 373

Genomic Informatics

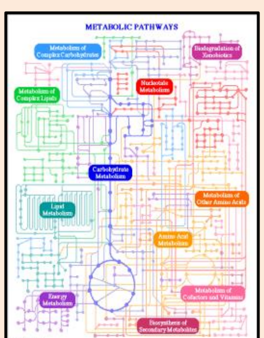
Elhanan Borenstein

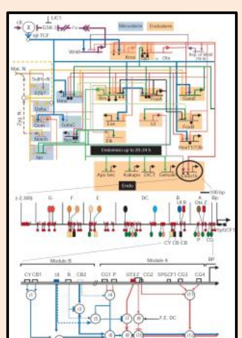
# A quick review

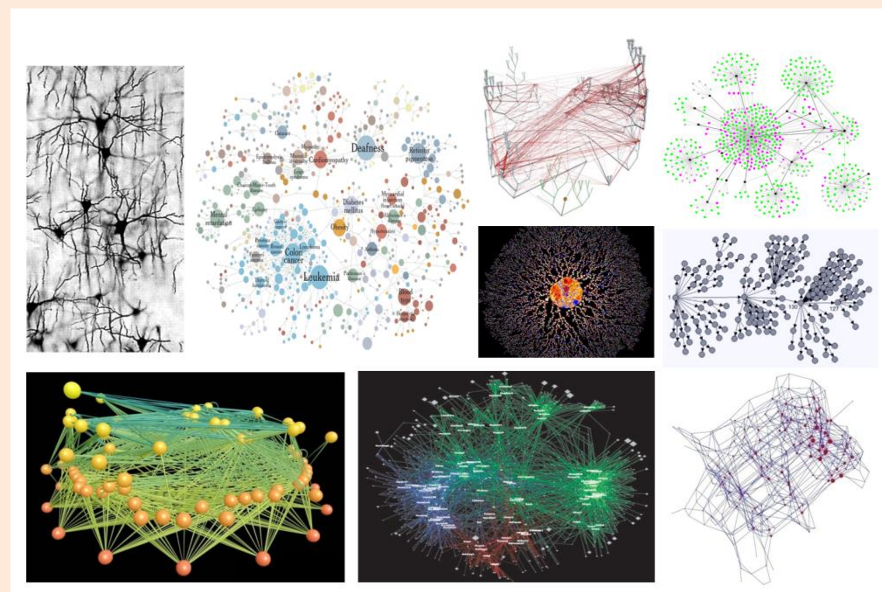
- Networks:
  - Networks vs. graphs
  - The Seven Bridges of Königsberg
  - A collection of **nodes** and **links**
  - Directed/undirected; weighted/non-weighted, …



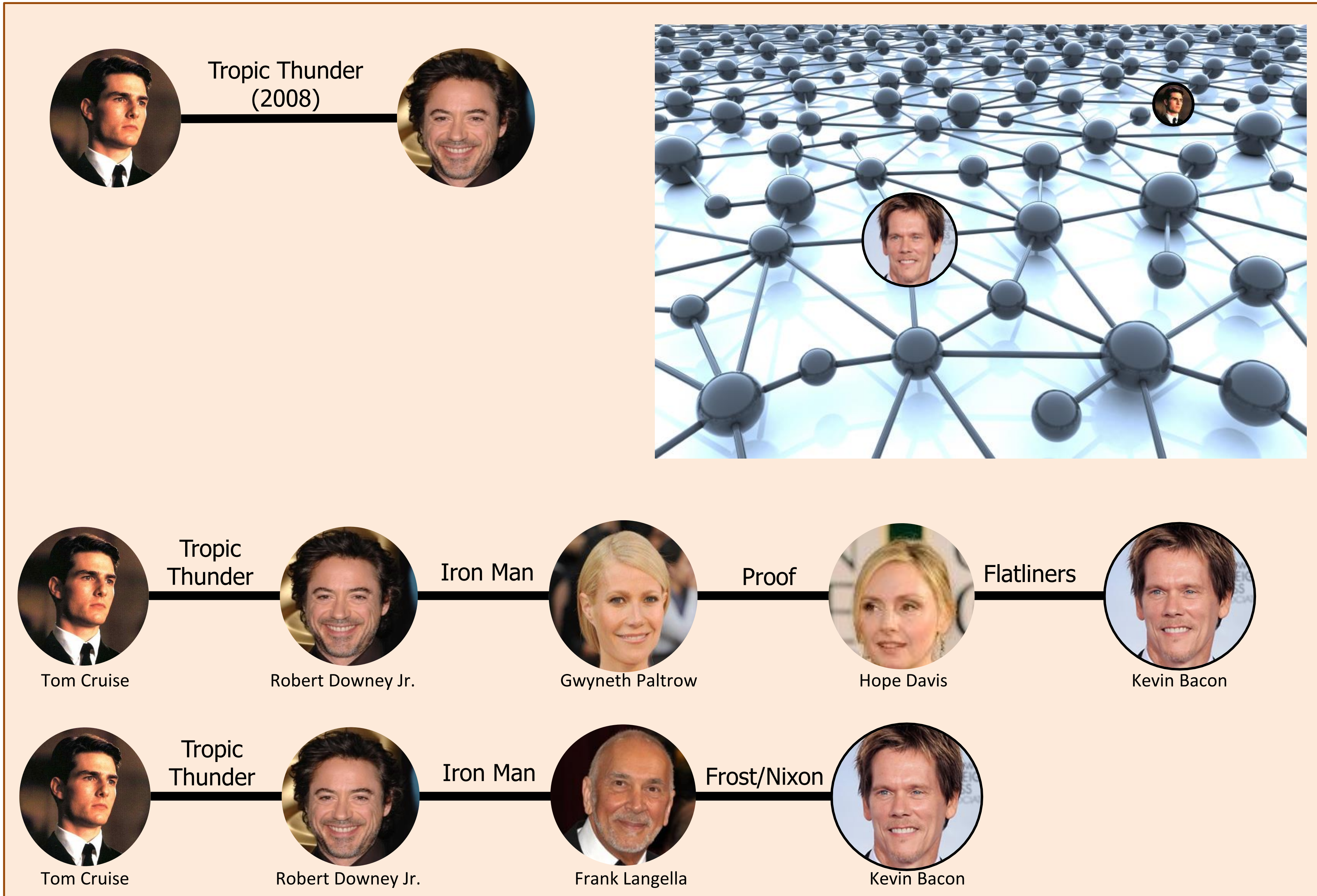

- Many types of biological networks
  - Transcriptional regulatory networks
  - Metabolic networks
  - Protein-protein interaction (PPI) networks



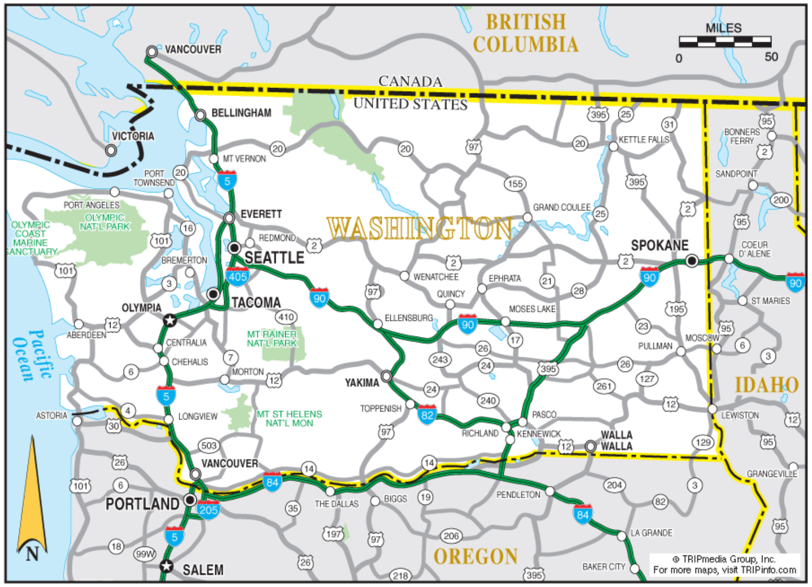


# The Bacon Number Game

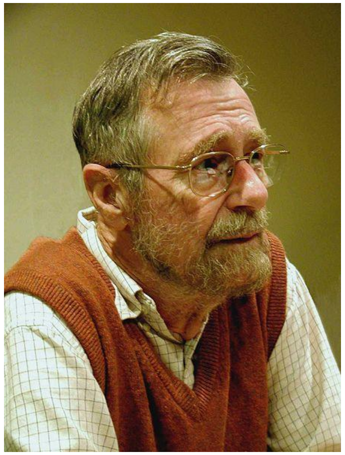


# The shortest path problem

- **Find the minimal number of "links" connecting node A to node B in an undirected network**
  - How many friends between you and someone on FB (6 degrees of separation, Erdös number, Kevin Bacon number)
  - How far apart are two genes in an interaction network
  - What is the shortest (and likely) infection path
- **Find the shortest (cheapest) path between two nodes in a weighted directed graph**
  - GPS; Google map



# Dijkstra’s Algorithm

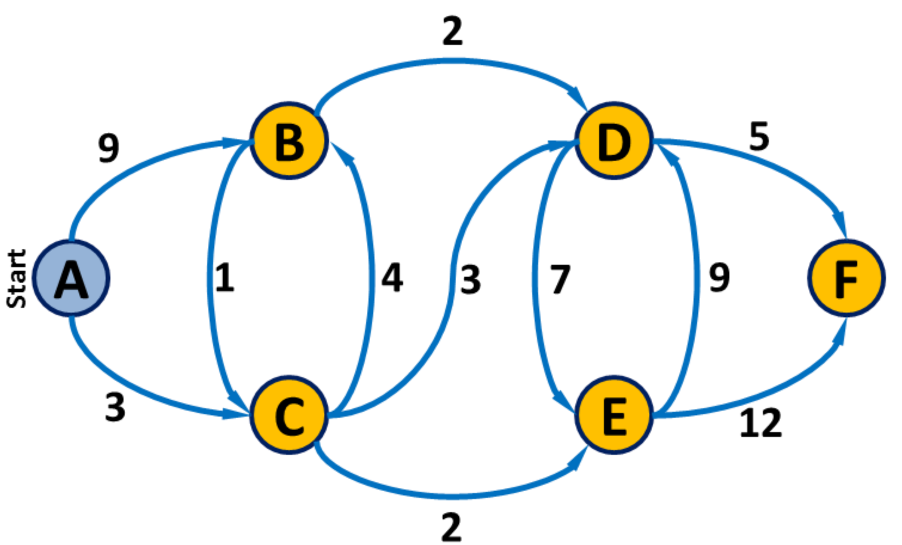


Edsger Wybe Dijkstra
1930 –2002

*"Computer Science is no more about computers than astronomy is about telescopes."*

# Dijkstra's algorithm

- **Solves the single-source shortest path problem:**
  - Find the shortest path from a single source to **ALL** nodes in the network
  - Works on both **directed** and **undirected** networks
  - Works on both **weighted** and **non-weighted** networks
- **Approach:**
  - Maintain shortest path to each intermediate node
- **Greedy algorithm**
  - … but still guaranteed to provide optimal solution !!

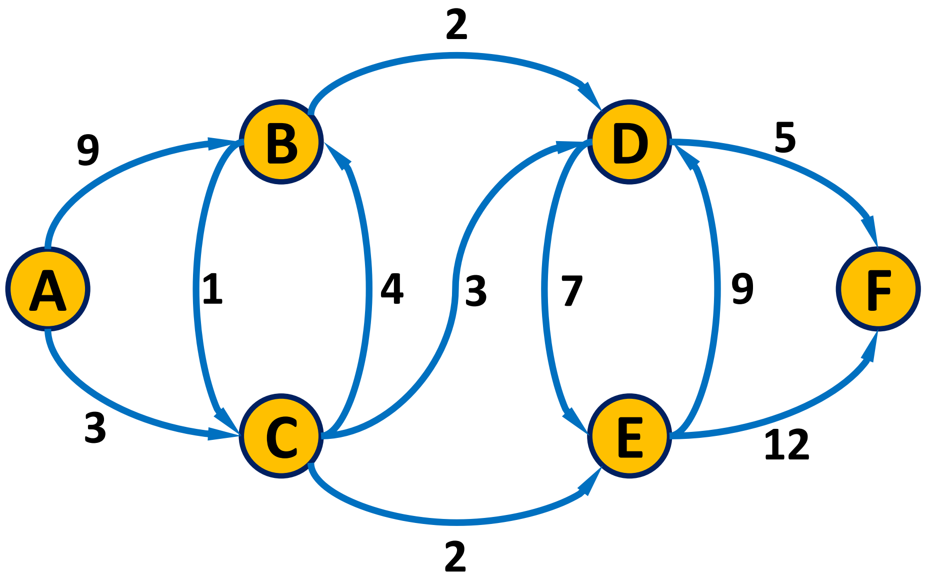


# Dijkstra's algorithm

1. **Initialize**:
   i. Assign a distance value, D, to each node.
      Set D to zero for *start* node and to infinity for all others.
   ii. Mark all nodes as unvisited.
   iii. Set *start* node as current node.
2. **For each of the current node's unvisited neighbors:**
   i. Calculate tentative distance, $D^t$, through current node.
   ii. If $D^t$ smaller than D (previously recorded distance): D← $D^t$
   iii. Mark current node as visited (note: shortest dist. found).
3. **Set the unvisited node with the smallest distance as the next "current node" and continue from step 2.**
4. Once all nodes are marked as visited, finish.

# Dijkstra's algorithm

- A simple synthetic network

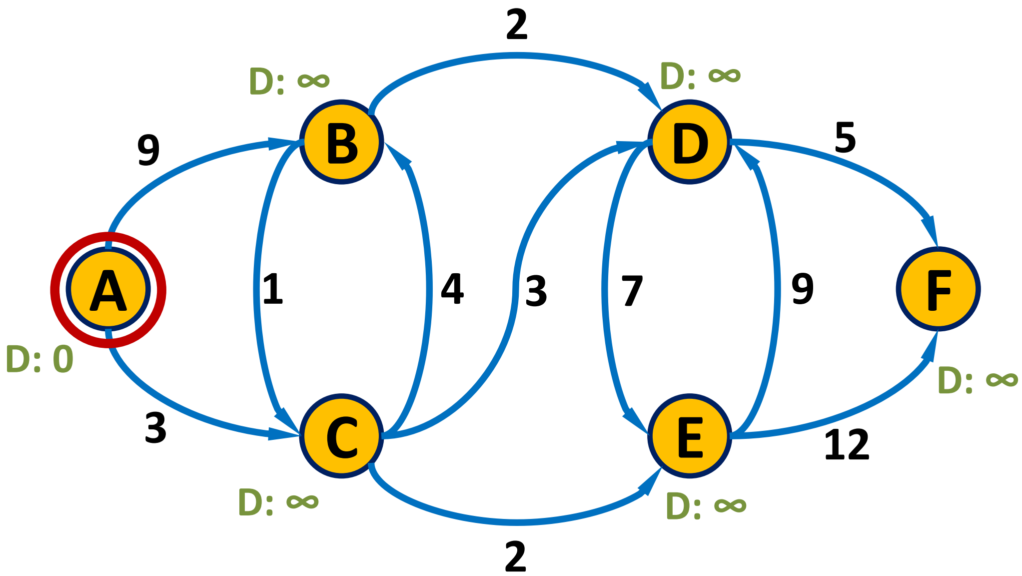



1. Initialize:
   i. Assign a distance value, D, to each node.
   Set D to zero for *start* node and to infinity for all others.
   ii. Mark all nodes as unvisited.
   iii. Set *start* node as current node.
2. For each of the current node's unvisited neighbors:
   i. Calculate tentative distance, $D^t$, through current node.
   ii. If $D^t$ smaller than D (previously recorded distance): D $\leftarrow$ $D^t$
   iii. Mark current node as visited (note: shortest dist. found).
3. Set the unvisited node with the smallest distance as the next "current node" and continue from step 2.
4. Once all nodes are marked as visited, finish.

# Dijkstra's algorithm

- Initialization
- Mark A (start) as current node

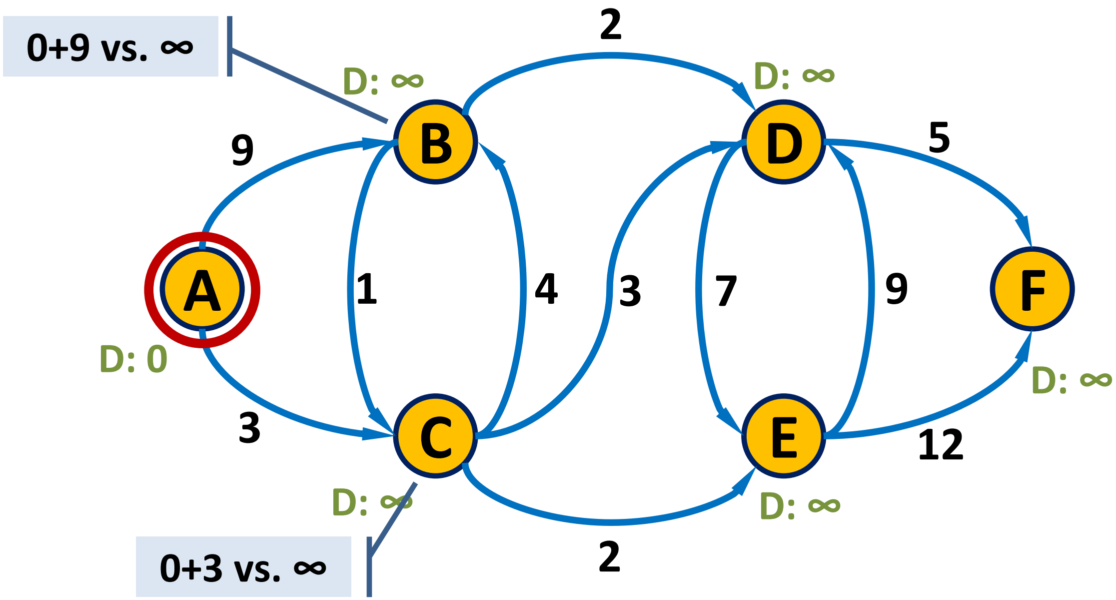
| A | B | C | D | E | F |
|---|---|---|---|---|---|
| 0 | ∞ | ∞ | ∞ | ∞ | ∞ |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |



# Dijkstra's algorithm

- Check unvisited neighbors of A

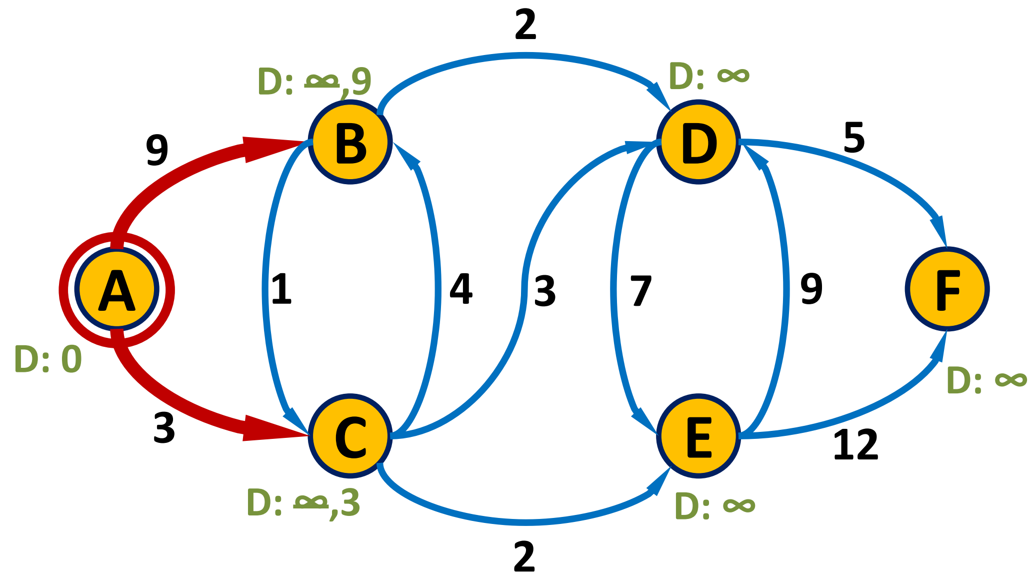
| A | B | C | D | E | F |
|---|---|---|---|---|---|
| 0 | ∞ | ∞ | ∞ | ∞ | ∞ |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |



# Dijkstra's algorithm

- Update D
- Record path

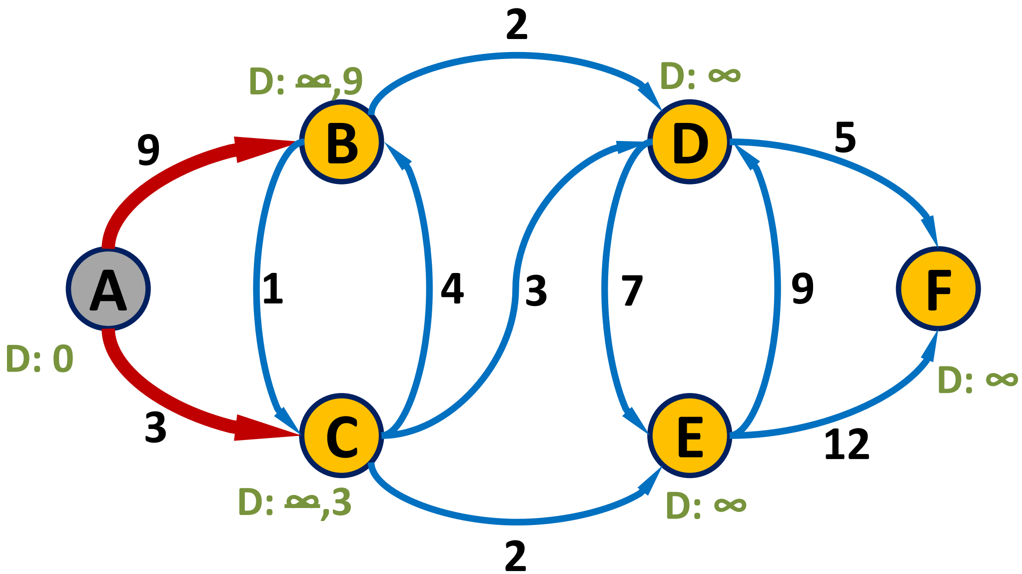
| A | B | C | D | E | F |
|---|---|---|---|---|---|
| 0 | ∞ | ∞ | ∞ | ∞ | ∞ |
| 0 | 9 | 3 | ∞ | ∞ | ∞ |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |



# Dijkstra's algorithm

- Mark A as visited …

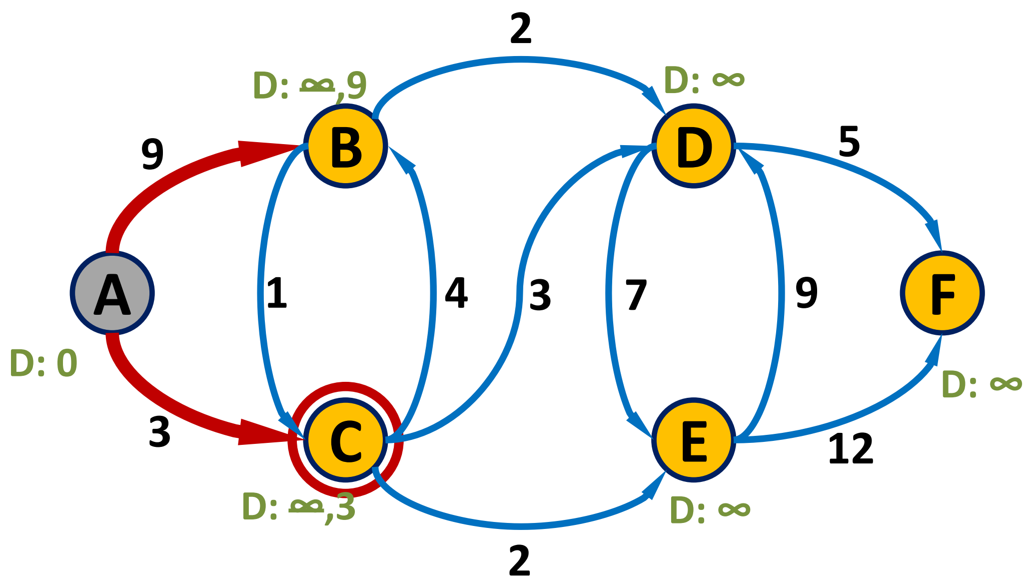
| A | B | C | D | E | F |
|---|---|---|---|---|---|
| 0 | ∞ | ∞ | ∞ | ∞ | ∞ |
| 0 | 9 | 3 | ∞ | ∞ | ∞ |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |



# Dijkstra's algorithm

- Mark C as current (unvisited node with smallest D)

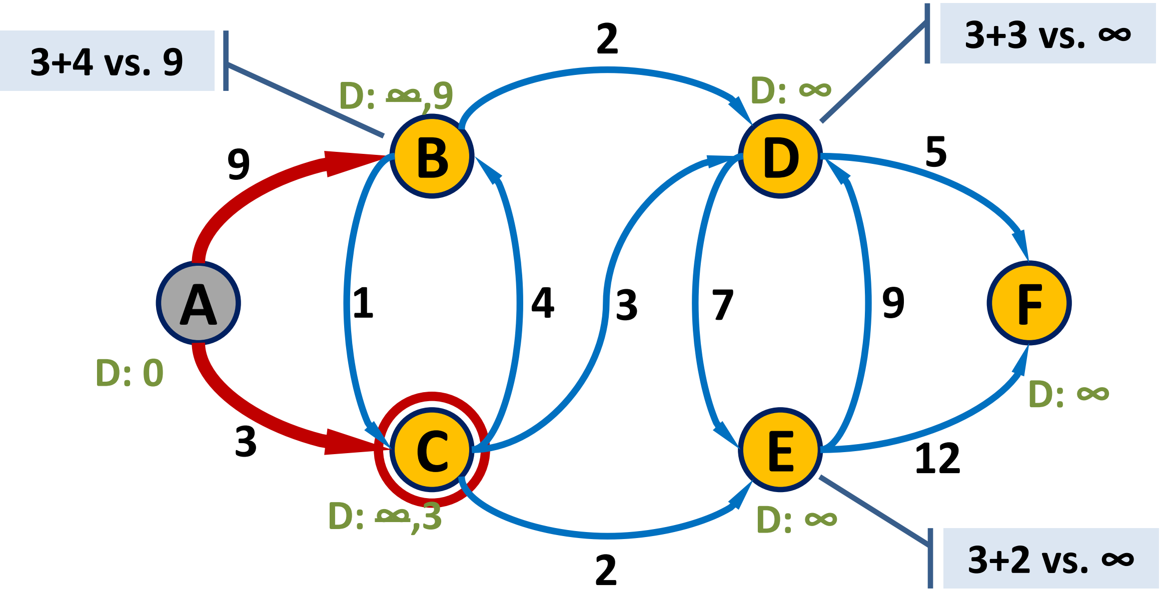
| A | B | C | D | E | F |
|---|---|---|---|---|---|
| 0 | ∞ | ∞ | ∞ | ∞ | ∞ |
| 0 | 9 | 3 | ∞ | ∞ | ∞ |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |



# Dijkstra’s algorithm

- Check unvisited neighbors of C

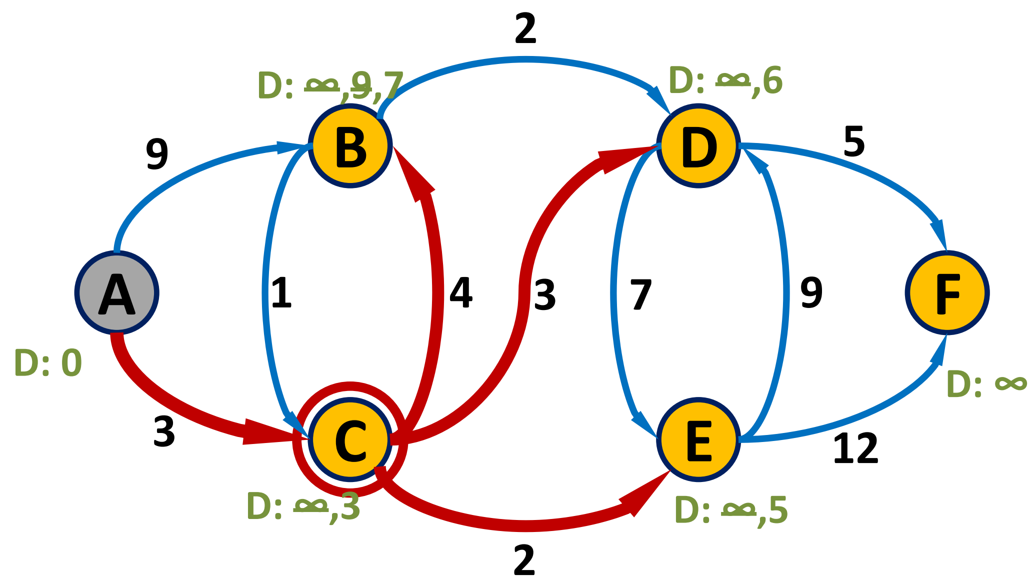
| A | B | C | D | E | F |
|---|---|---|---|---|---|
| 0 | ∞ | ∞ | ∞ | ∞ | ∞ |
| 0 | 9 | 3 | ∞ | ∞ | ∞ |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |



# Dijkstra's algorithm

- Update distance
- Record path

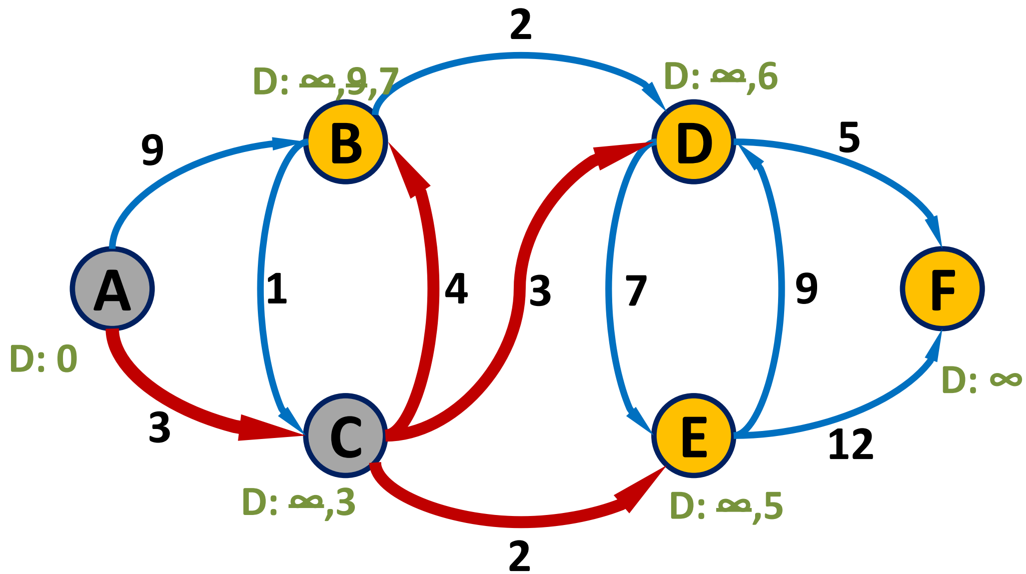
| A | B | C | D | E | F |
|---|---|---|---|---|---|
| 0 | ∞ | ∞ | ∞ | ∞ | ∞ |
| 0 | 9 | 3 | ∞ | ∞ | ∞ |
| | 7 | 3 | 6 | 5 | ∞ |
| | | | | | |
| | | | | | |
| | | | | | |



# Dijkstra's algorithm

- Mark C as visited
- Note: Distance to C is final!!

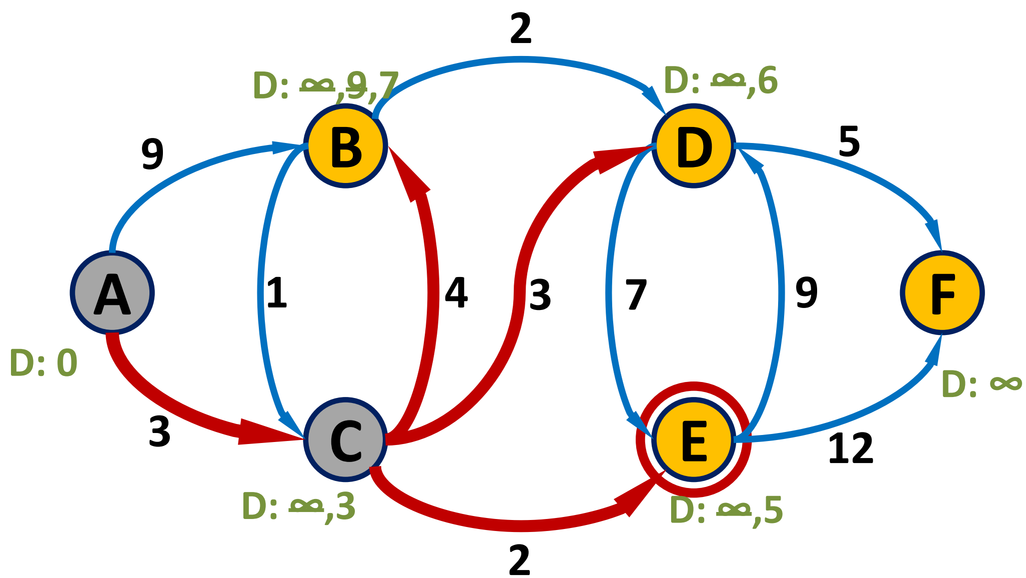
| A | B | C | D | E | F |
|---|---|---|---|---|---|
| 0 | ∞ | ∞ | ∞ | ∞ | ∞ |
| 0 | 9 | 3 | ∞ | ∞ | ∞ |
| | 7 | 3 | 6 | 5 | ∞ |
| | | | | | |
| | | | | | |
| | | | | | |



# Dijkstra's algorithm

- Mark E as current node
- Check unvisited neighbors of E

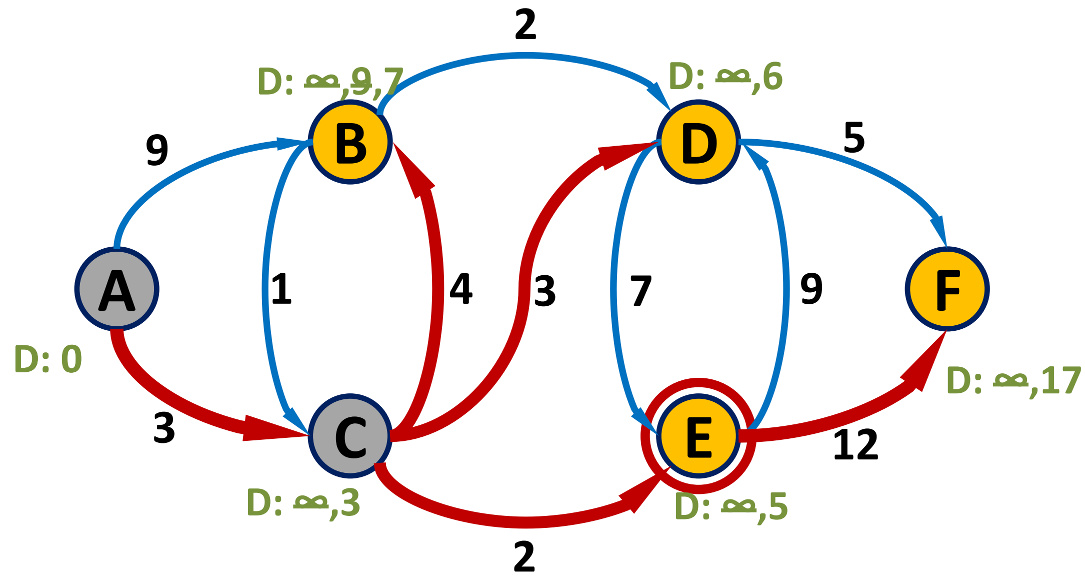
| A | B | C | D | E | F |
|---|---|---|---|---|---|
| 0 | ∞ | ∞ | ∞ | ∞ | ∞ |
| 0 | 9 | 3 | ∞ | ∞ | ∞ |
| | 7 | 3 | 6 | 5 | ∞ |
| | | | | | |
| | | | | | |
| | | | | | |



# Dijkstra's algorithm

- Update D
- Record path

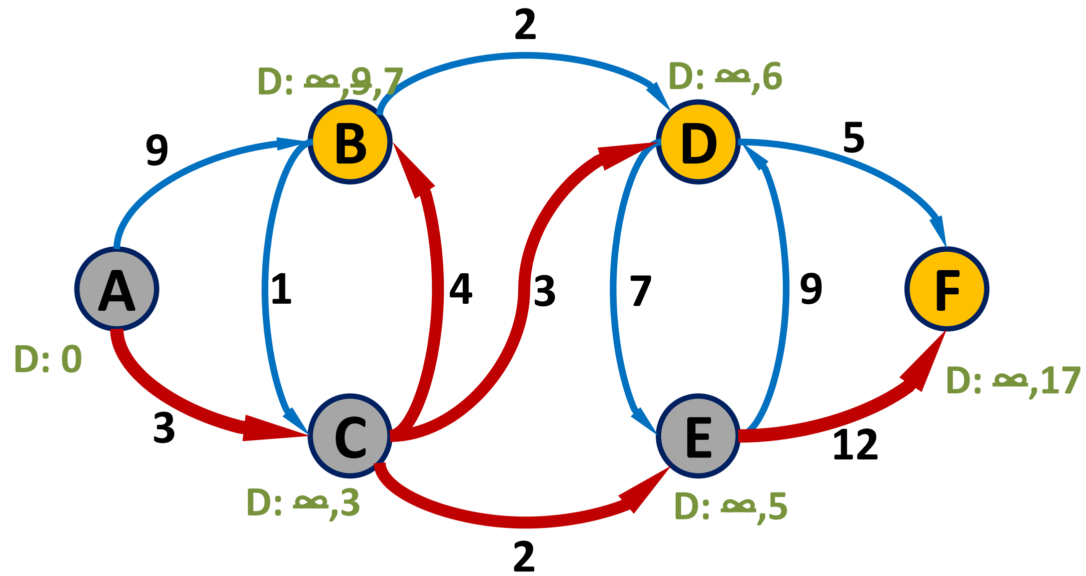
| A | B | C | D | E | F |
|---|---|---|---|---|---|
| 0 | ∞ | ∞ | ∞ | ∞ | ∞ |
| 0 | 9 | 3 | ∞ | ∞ | ∞ |
| | 7 | 3 | 6 | 5 | ∞ |
| | 7 | | 6 | 5 | 17 |
| | | | | | |
| | | | | | |



# Dijkstra's algorithm

- Mark E as visited

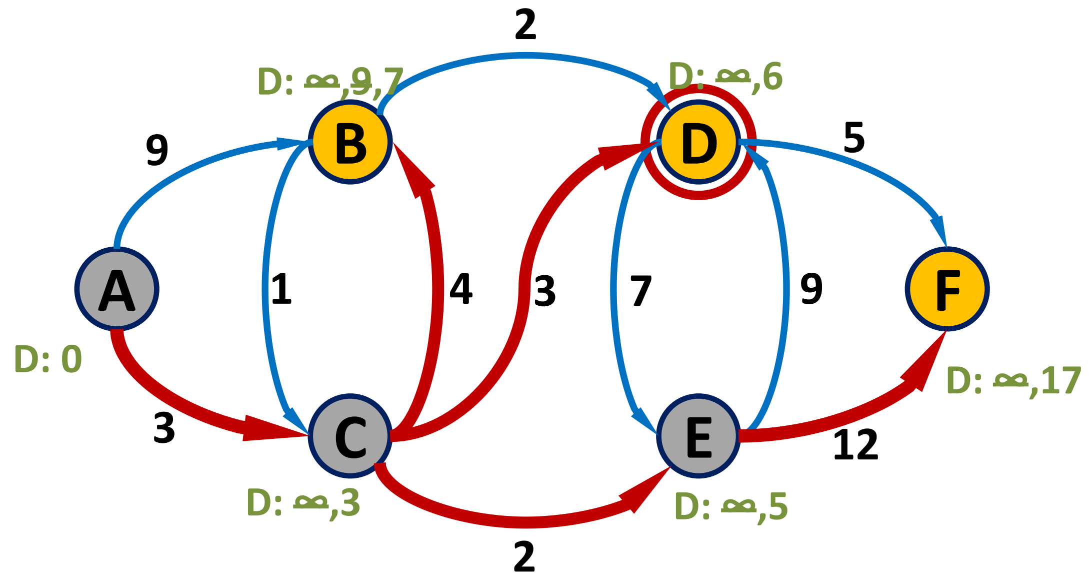
| A | B | C | D | E | F |
|---|---|---|---|---|---|
| 0 | ∞ | ∞ | ∞ | ∞ | ∞ |
| 0 | 9 | 3 | ∞ | ∞ | ∞ |
| | 7 | 3 | 6 | 5 | ∞ |
| | 7 | | 6 | 5 | 17 |
| | | | | | |
| | | | | | |



# Dijkstra's algorithm

- Mark D as current node
- Check unvisited neighbors of D

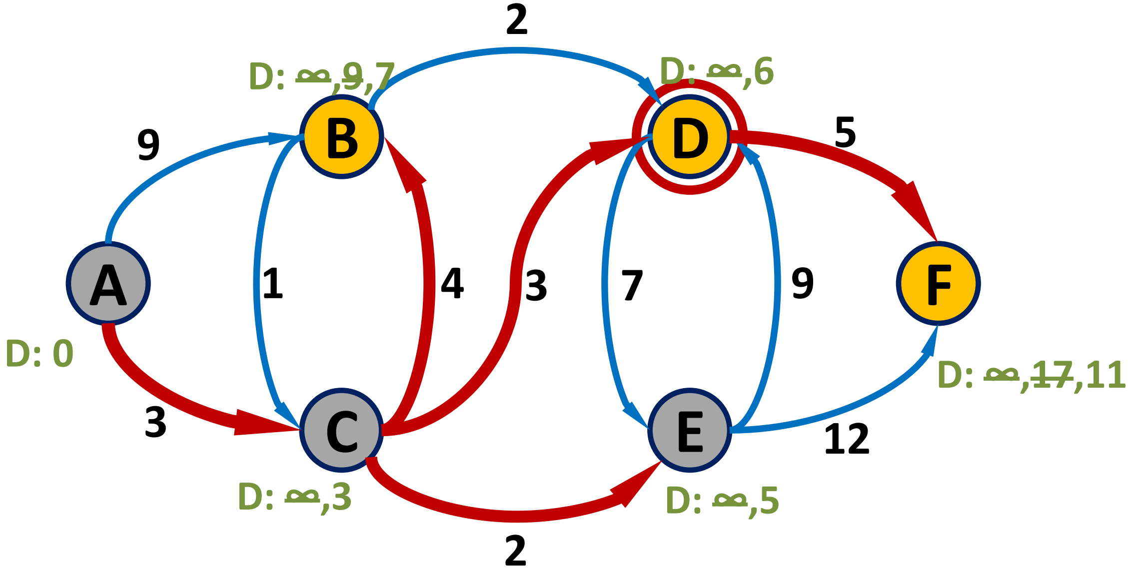
| A | B | C | D | E | F |
|---|---|---|---|---|---|
| 0 | ∞ | ∞ | ∞ | ∞ | ∞ |
| 0 | 9 | 3 | ∞ | ∞ | ∞ |
| | 7 | 3 | 6 | 5 | ∞ |
| | 7 | | 6 | 5 | 17 |
| | | | | | |
| | | | | | |



# Dijkstra's algorithm

- Update D
- Record path (note: path has changed)

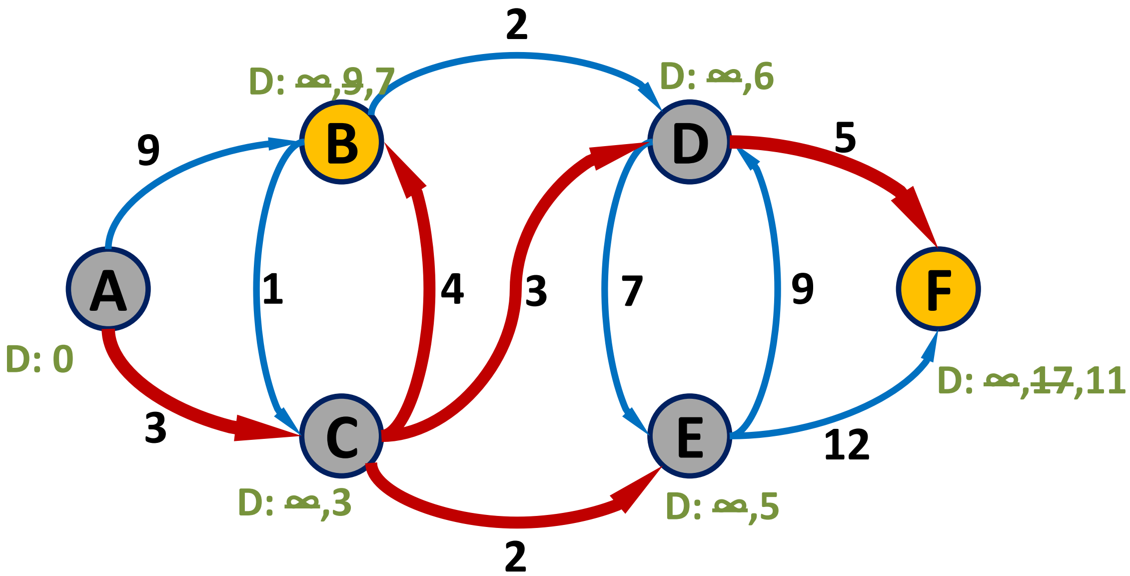
| A | B | C | D | E | F |
|---|---|---|---|---|---|
| 0 | ∞ | ∞ | ∞ | ∞ | ∞ |
| 0 | 9 | 3 | ∞ | ∞ | ∞ |
| | 7 | 3 | 6 | 5 | ∞ |
| | 7 | | 6 | 5 | 17 |
| | 7 | | 6 | | 11 |
| | | | | | |



# Dijkstra’s algorithm

- Mark D as visited

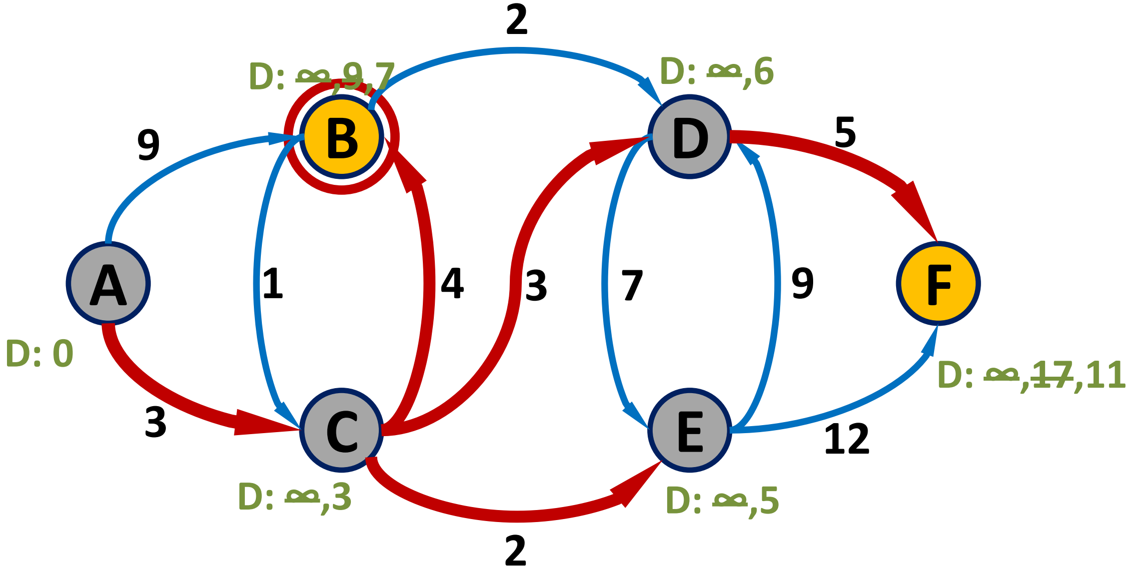
| A | B | C | D | E | F |
|---|---|---|---|---|---|
| 0 | ∞ | ∞ | ∞ | ∞ | ∞ |
| 0 | 9 | 3 | ∞ | ∞ | ∞ |
| | 7 | 3 | 6 | 5 | ∞ |
| | 7 | | 6 | 5 | 17 |
| | 7 | | 6 | | 11 |
| | | | | | |



# Dijkstra's algorithm

- Mark B as current node
- Check neighbors

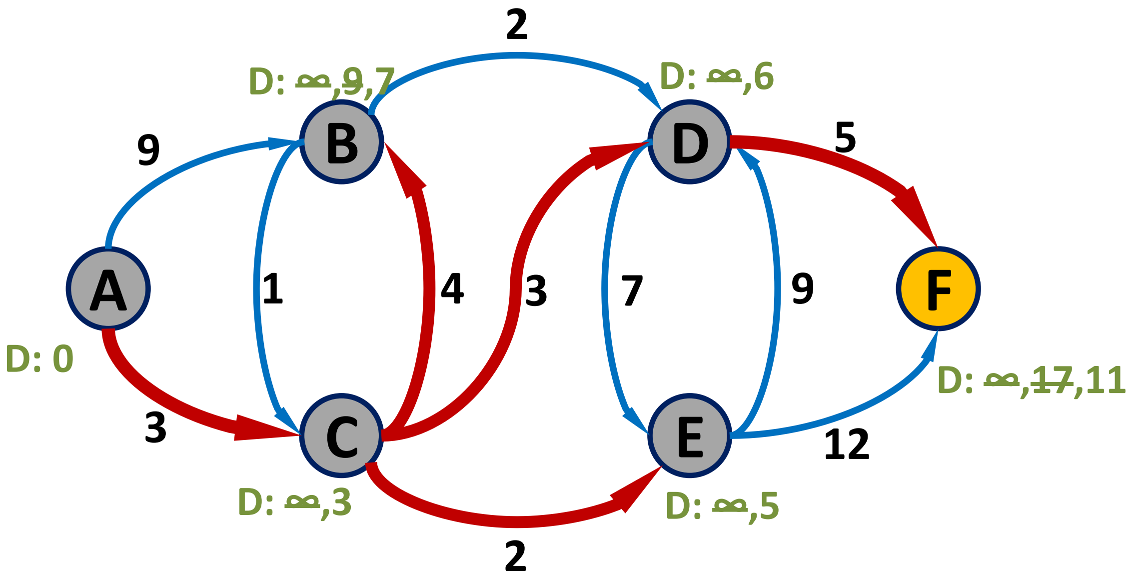
| A | B | C | D | E | F |
|---|---|---|---|---|---|
| 0 | ∞ | ∞ | ∞ | ∞ | ∞ |
| 0 | 9 | 3 | ∞ | ∞ | ∞ |
| | 7 | 3 | 6 | 5 | ∞ |
| | 7 | | 6 | 5 | 17 |
| | 7 | | 6 | | 11 |
| | | | | | |



# Dijkstra’s algorithm

- No updates..
- Mark B as visited

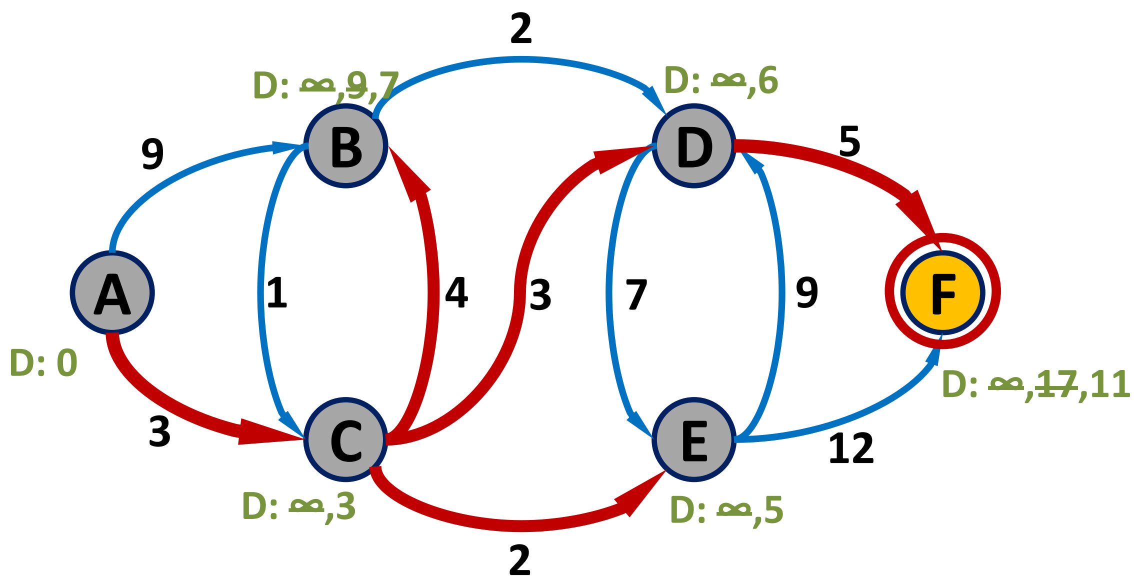
| A | B | C | D | E | F |
|---|---|---|---|---|---|
| 0 | ∞ | ∞ | ∞ | ∞ | ∞ |
| 0 | 9 | 3 | ∞ | ∞ | ∞ |
| | 7 | 3 | 6 | 5 | ∞ |
| | 7 | | 6 | 5 | 17 |
| | 7 | | 6 | | 11 |
| | 7 | | | | 11 |



# Dijkstra's algorithm

- Mark F as current

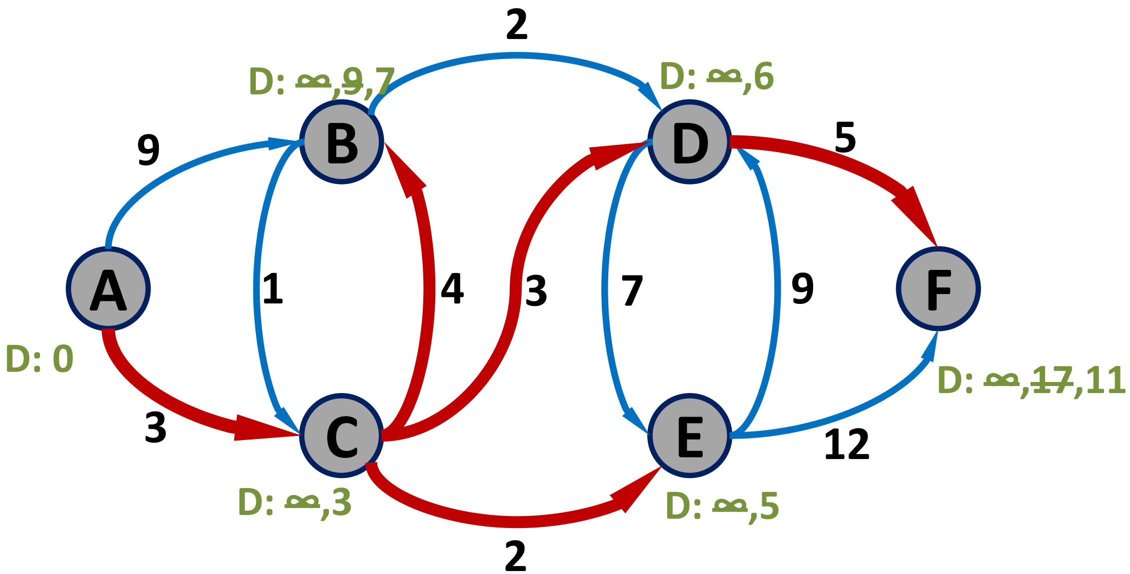
| A | B | C | D | E | F |
|---|---|---|---|---|---|
| 0 | ∞ | ∞ | ∞ | ∞ | ∞ |
| 0 | 9 | 3 | ∞ | ∞ | ∞ |
| | 7 | 3 | 6 | 5 | ∞ |
| | 7 | | 6 | 5 | 17 |
| | 7 | | 6 | | 11 |
| | 7 | | | | 11 |



# Dijkstra's algorithm

- Mark F as visited

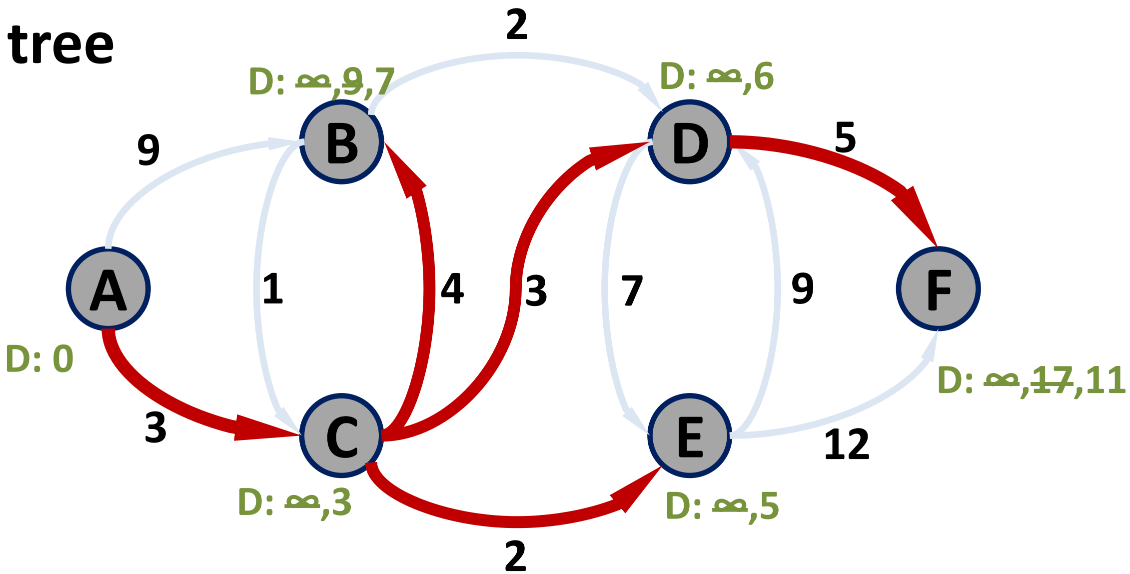
| A | B | C | D | E | F |
|---|---|---|---|---|---|
| 0 | ∞ | ∞ | ∞ | ∞ | ∞ |
| 0 | 9 | 3 | ∞ | ∞ | ∞ |
| | 7 | 3 | 6 | 5 | ∞ |
| | 7 | | 6 | 5 | 17 |
| | 7 | | 6 | | 11 |
| | 7 | | | | 11 |
| | | | | | 11 |



# We are done!

- We now have:
  - Shortest path from A to each node (both length and path)
  - **Minimum spanning tree**

| A | B | C | D | E | F |
|---|---|---|---|---|---|
| 0 | ∞ | ∞ | ∞ | ∞ | ∞ |
| 0 | 9 | 3 | ∞ | ∞ | ∞ |
| | 7 | 3 | 6 | 5 | ∞ |
| | 7 | | 6 | 5 | 17 |
| | 7 | | 6 | | 11 |
| | 7 | | | | 11 |
| | | | | | 11 |

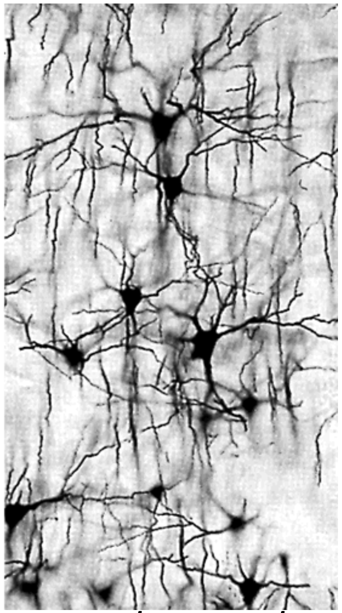



**Will we always get a tree?**

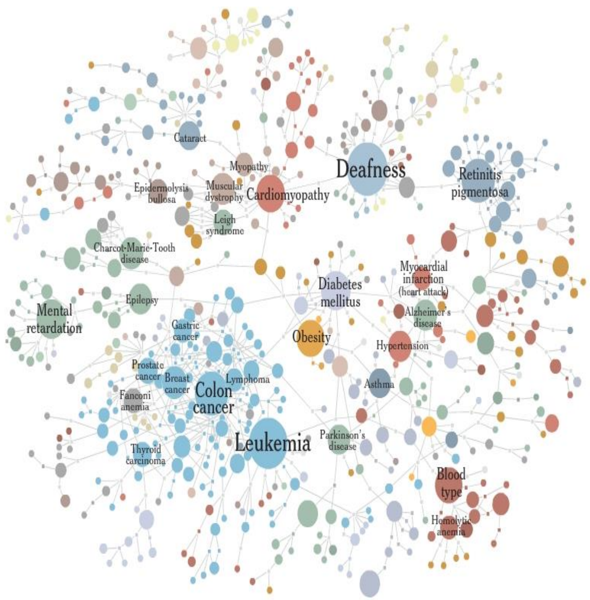
**Can you prove it?**

# Measuring Network Topology

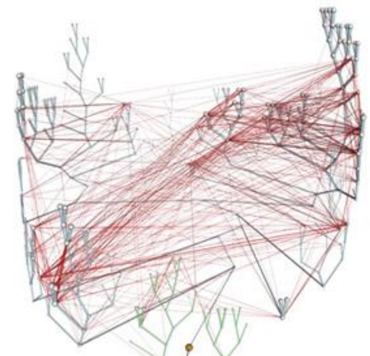
# Networks in biology/medicine



*neural network*

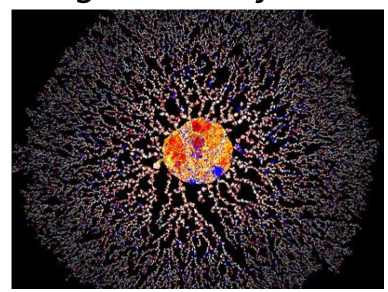


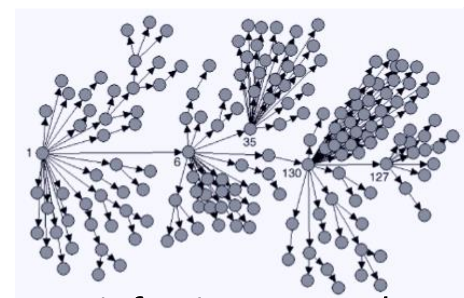

*disease network*



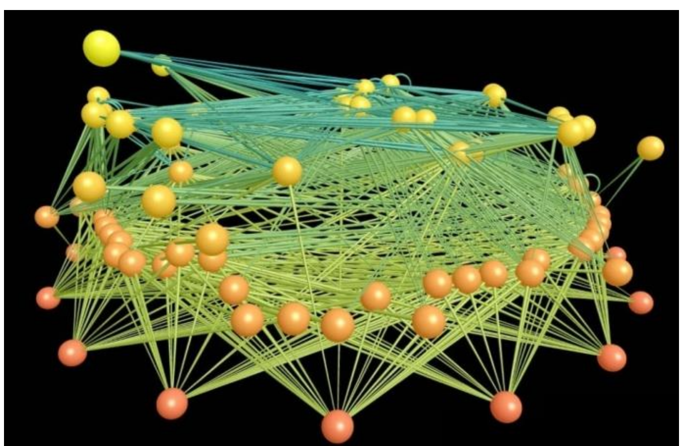
*gene transfer*

*social network*

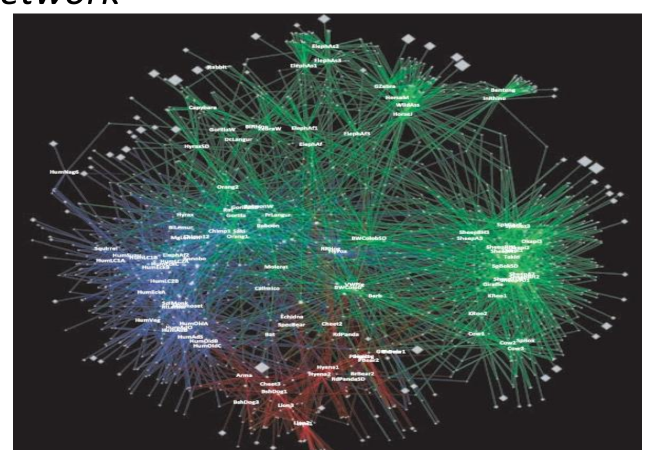




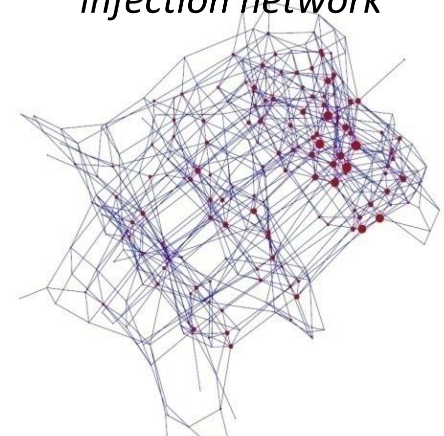

*infection network*



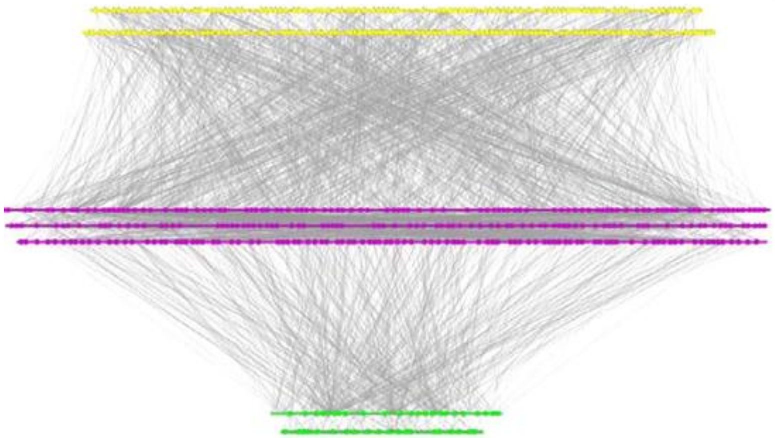
*food web*

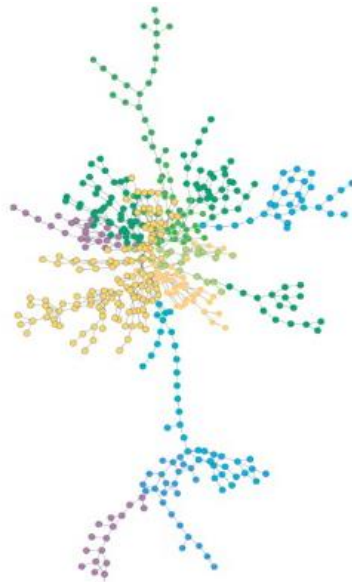


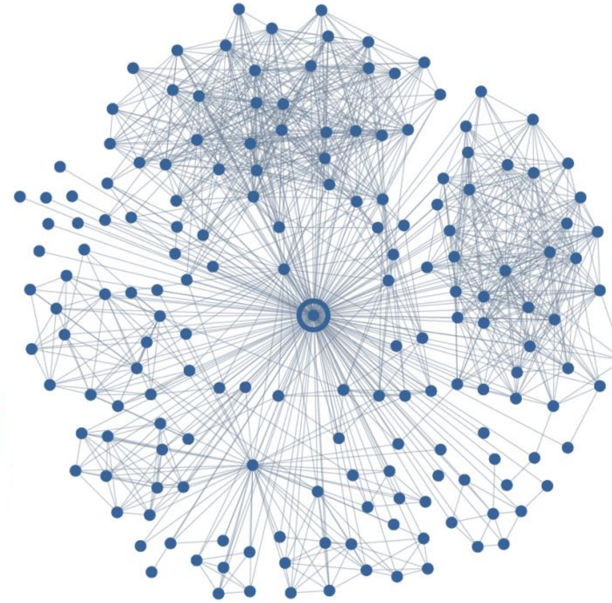
*microbial-host association network*

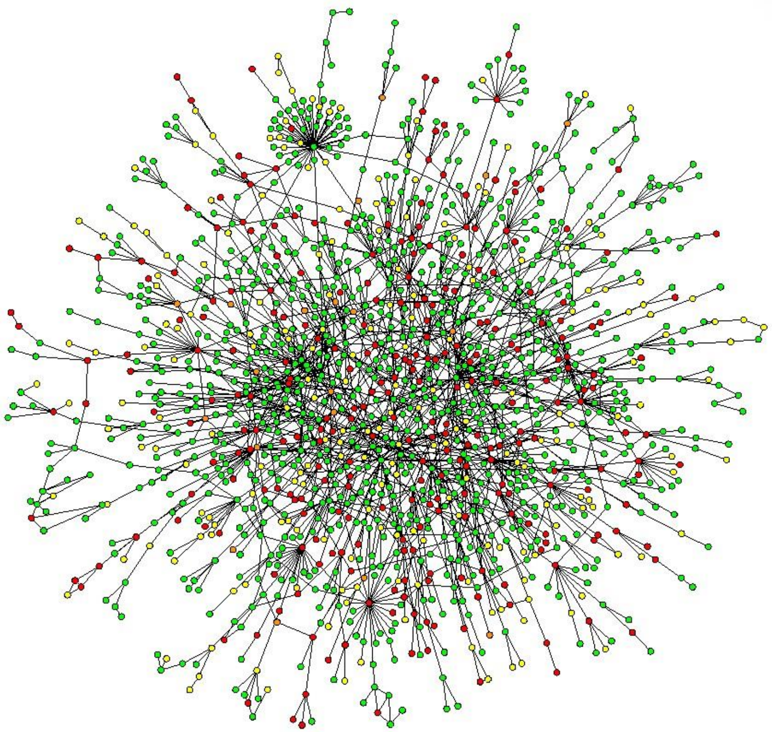


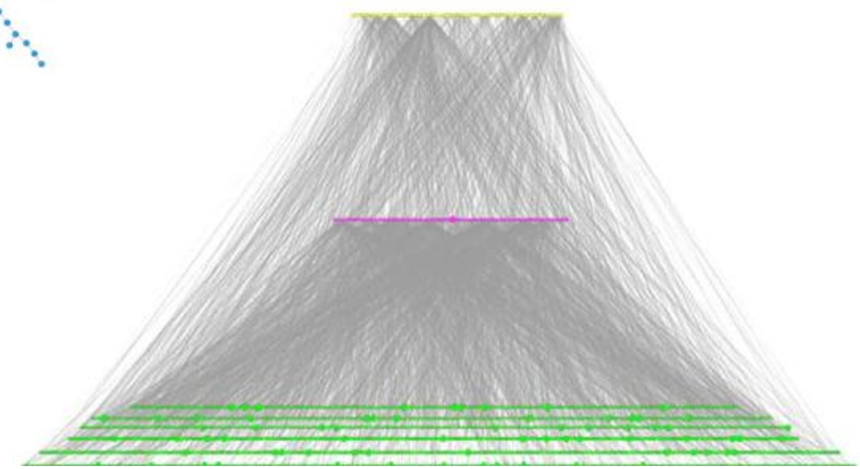
*mutation network*











# Comparing networks

- We want to find a way to "compare" networks.
  - "Similar" (not identical) **topology**
  - "Common" **design principles**

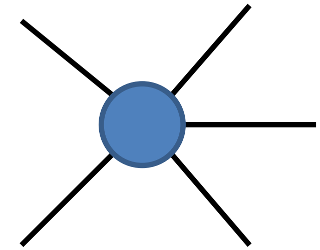
- We seek measures of network topology that are:
  - Simple
  - Capture **global** organization
  - Potentially "important"

  (equivalent to, for example, GC content for genomes)

**Summary statistics**

# Node degree / rank

- Degree = Number of neighbors

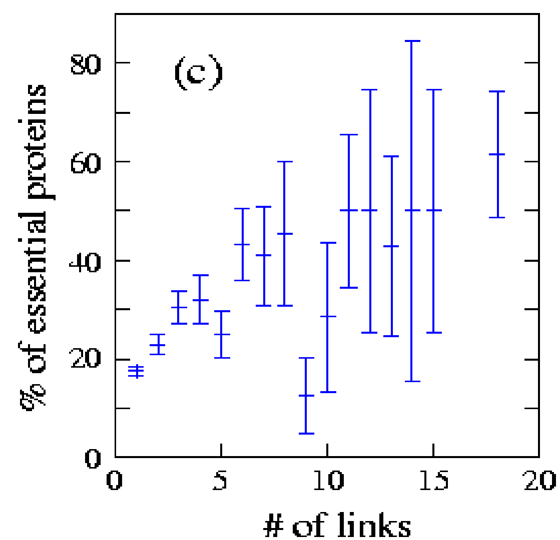


- Node degree in PPI networks correlates with:
  - Gene essentiality
  - Conservation rate
  - Likelihood to cause human disease

brief communications

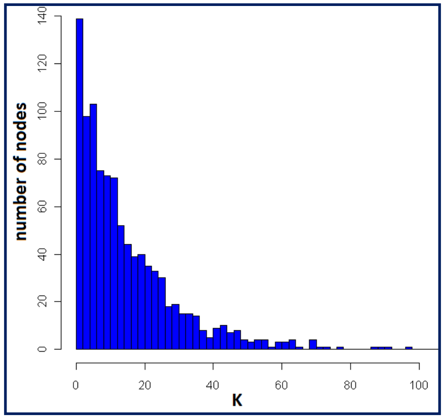
Lethality and centrality in protein networks

The most highly connected proteins in the cell are the most important for its survival.



# Degree distribution

- P(k): probability that a node has a degree of exactly k




- Potential distributions (and how they 'look'):

**Poisson:**

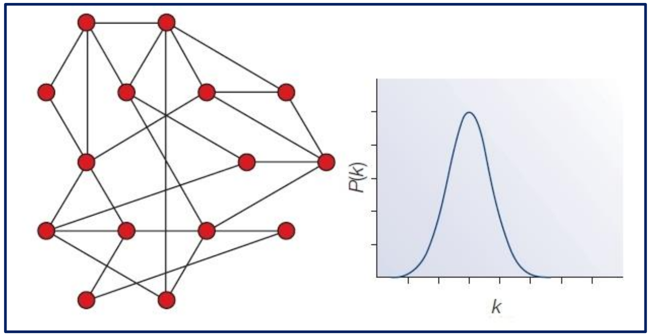
$$P(k) = \frac{e^{-d} d^k}{k!}$$

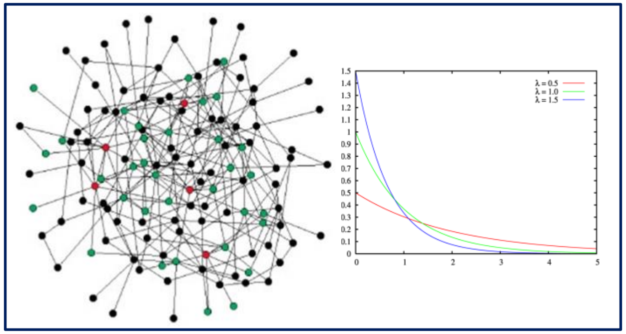
**Exponential:**

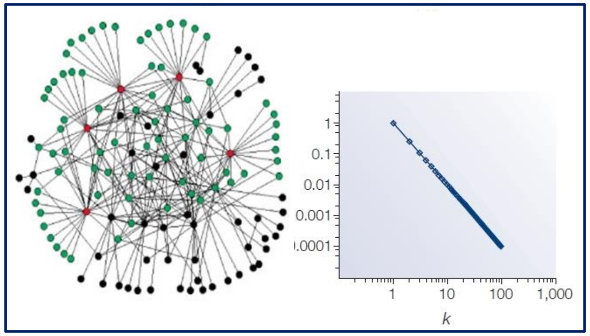
$$P(k) \propto e^{-k/d}$$

**Power-law:**

$$P(k) \propto k^{-c}, k \neq 0, c > 1$$



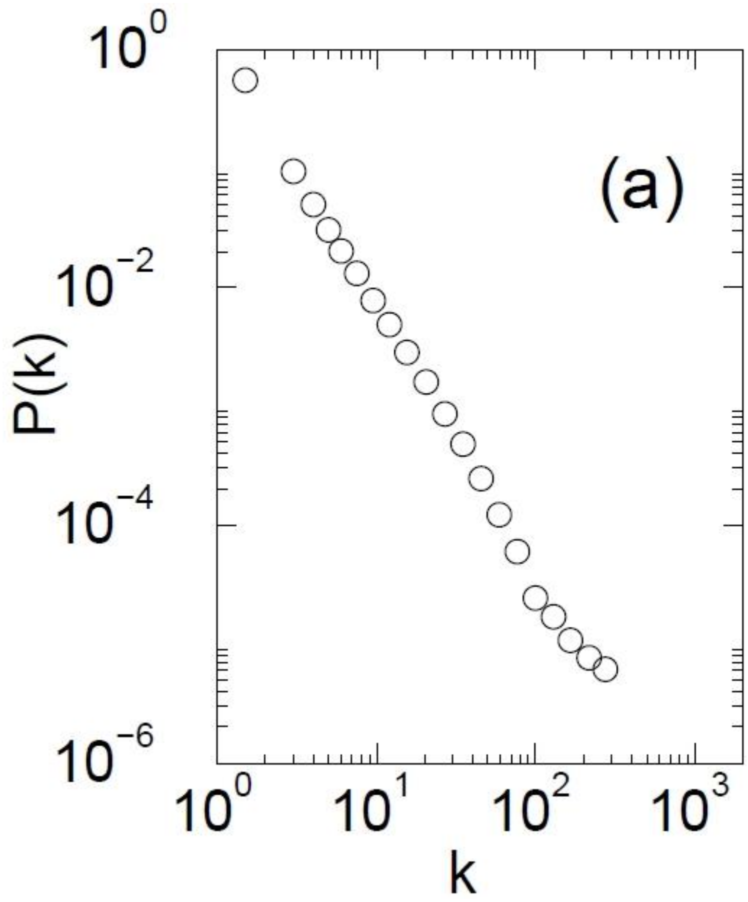





# The Internet

- **Nodes** – 150,000 routers
- **Edges** – physical links

- $P(k) \sim k^{-2.3}$

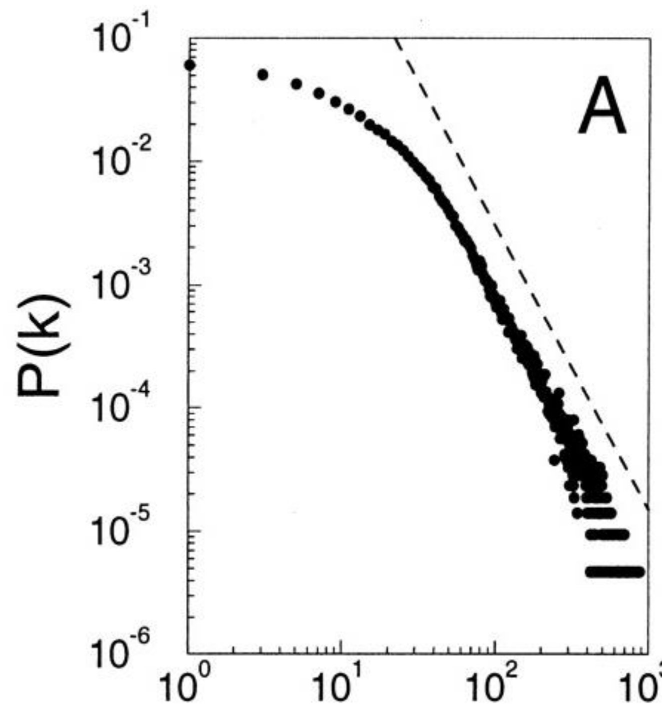



Govindan and Tangmunarunkit, 2000

# Movie actor collaboration network


- **Nodes** – 212,250 actors
- **Edges** – co-appearance in a movie

- $P(k) \sim k^{-2.3}$

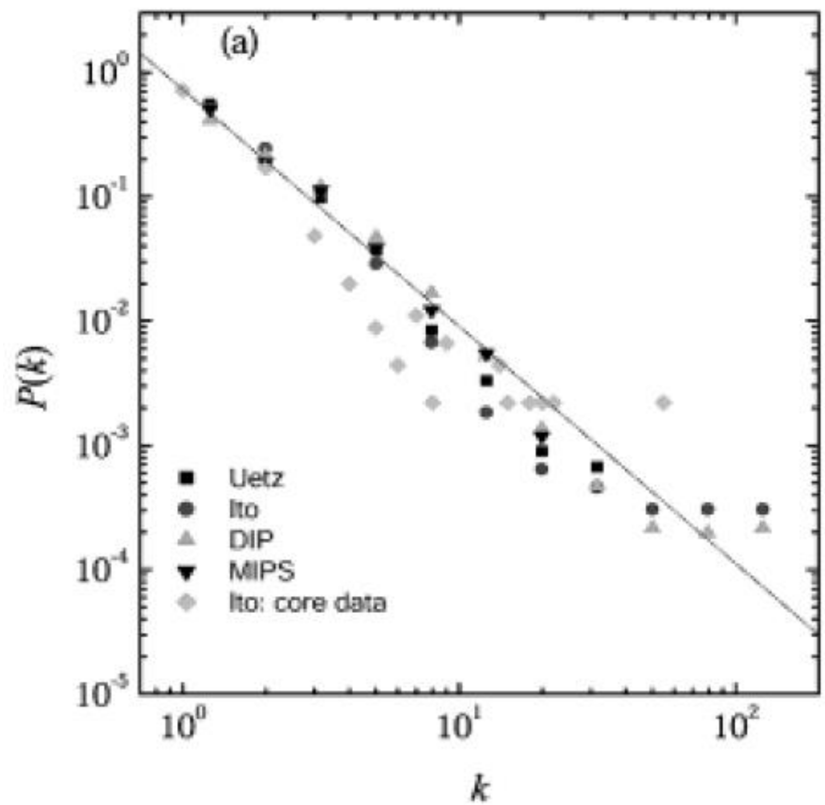



Barabasi and Albert, Science, 1999

# Protein protein interaction networks

- **Nodes** – Proteins
- **Edges** – Interactions (yeast)

- $P(k) \sim k^{-2.5}$



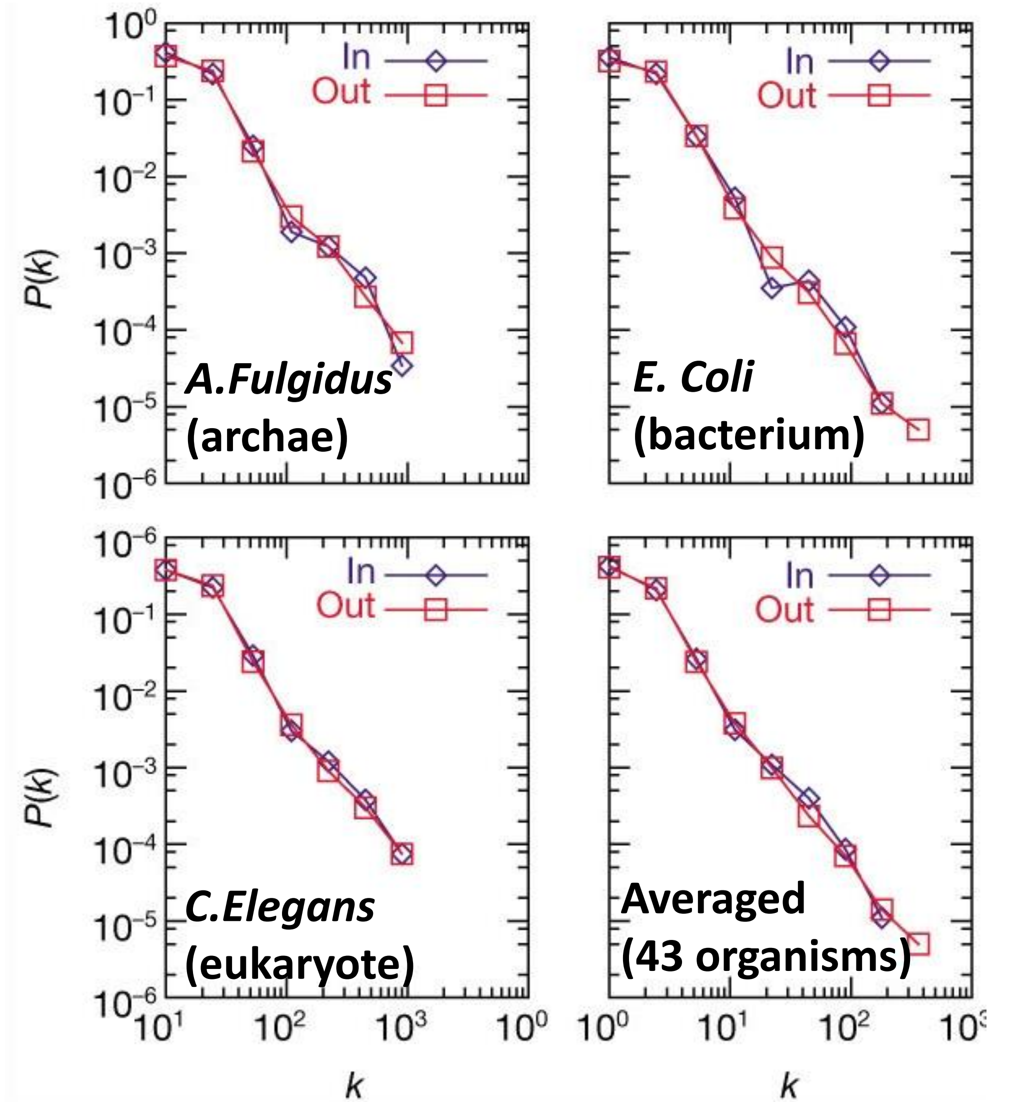

Yook et al, Proteomics, 2004

# Metabolic networks

- **Nodes** – Metabolites
- **Edges** – Reactions

- $P(k) \sim k^{-2.2\pm2}$

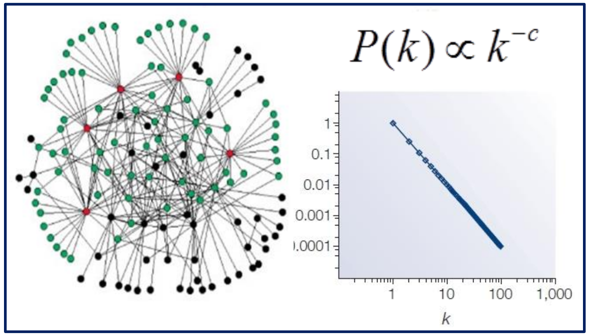
***Metabolic networks across all kingdoms of life are scale-free***




Jeong et al., Nature, 2000

# The power-law distribution

- **Power-law distribution has a “heavy” tail!**
  - Characterized by a small number of highly connected nodes, known as **hubs**
  - A.k.a. “scale-free” network




- **Hubs are crucial:**
  - Affect **error** and **attack** tolerance of complex networks (Albert et al. Nature, 2000)


# Why do so many real-life networks exhibit a power-law degree distribution?

- *Is it "selected for"?*
- *Is it expected by chance?*
- *Does it have anything to do with the way networks evolve?*
- *Does it have functional implications?*

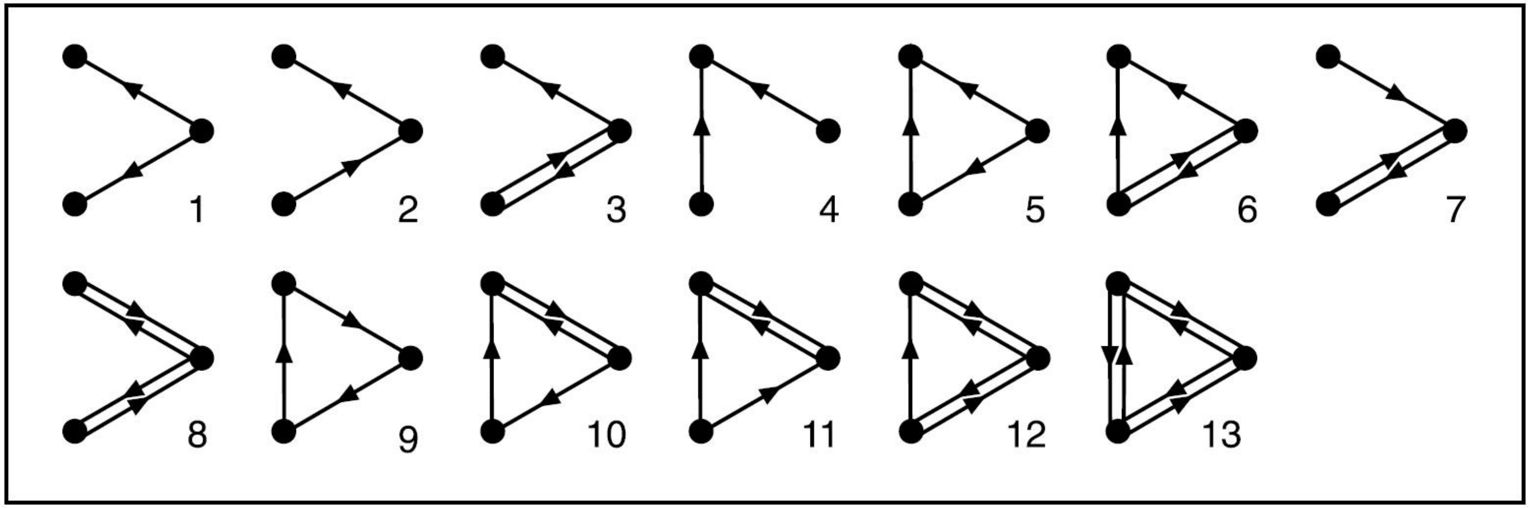
**?**

# Network Motifs

- Going beyond degree distribution …
- Generalization of sequence motifs
- Basic building blocks
- Evolutionary design principles?

# What are network motifs?

- Recurring patterns of interaction (***sub-graphs***) that are significantly **overrepresented** (w.r.t. a background model)




13 possible 3-nodes sub-graphs
(199 possible 4-node sub-graphs)

R. Milo et al. Network motifs: simple building blocks of complex networks. Science, 2002

# Finding motifs in the network

1a. Scan all n-node sub-graphs in the ***real*** network

1b. Record number of appearances of each sub-graph (*consider isomorphic architectures*)

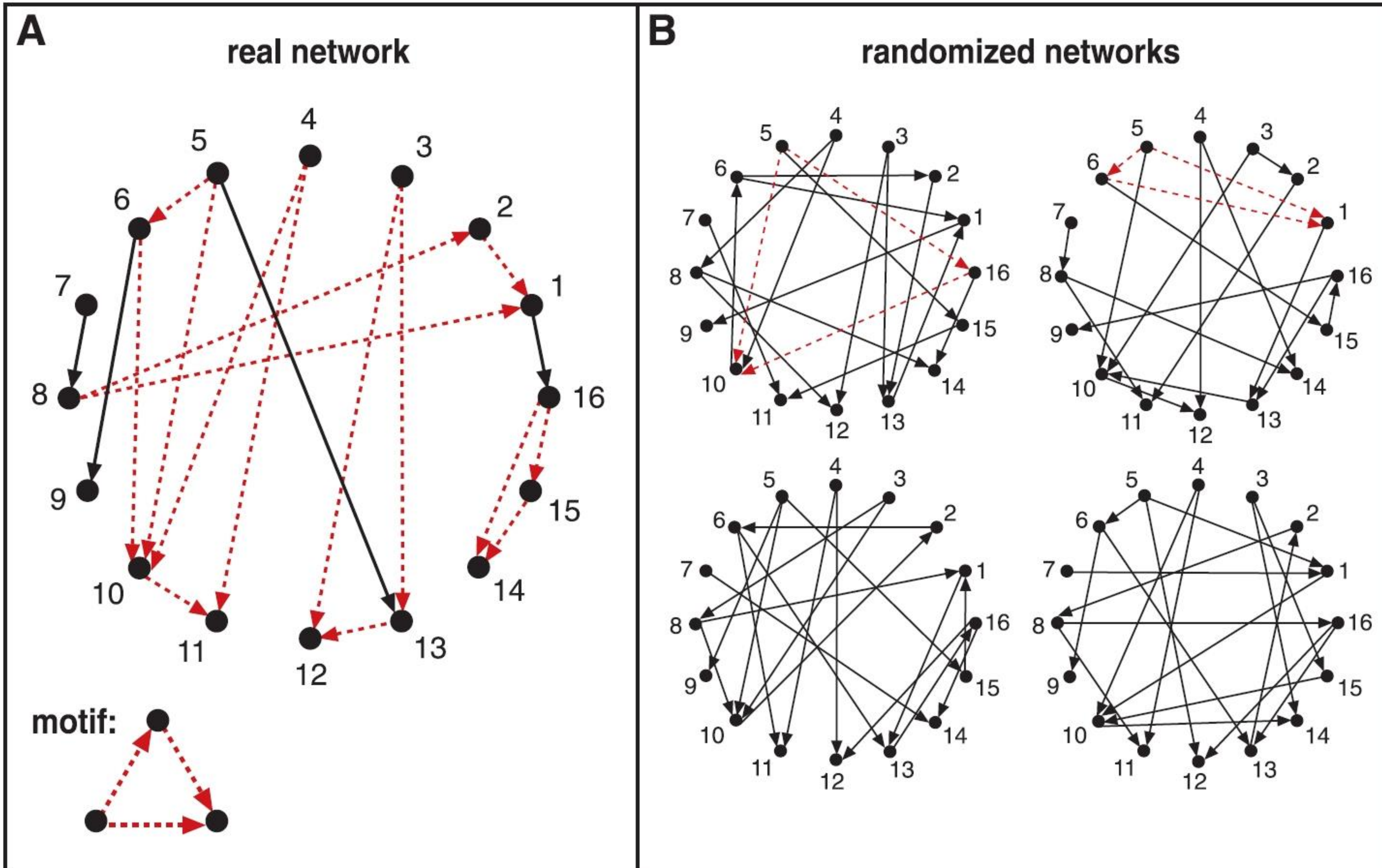
2. Generate a large set of random networks

3a. Scan for all n-node sub-graphs in **random** networks

3b. Record number of appearances of each sub-graph

4. Compare each sub-graph's data and identify motifs

# Finding motifs in the network



# Network randomization

- How should the set of random networks be generated?
- Do we really want “completely random” networks?
- What constitutes a good null model?

# Network randomization

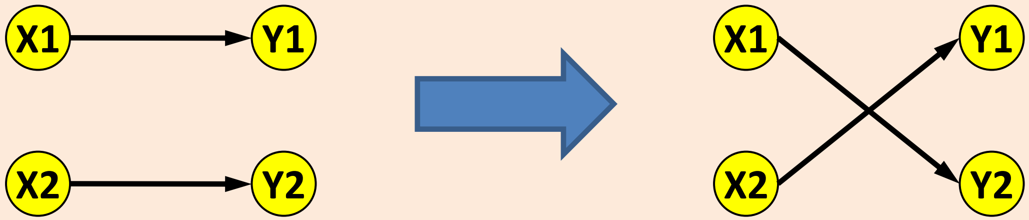
- How should the set of random networks be generated?
- Do we really want “completely random” networks?
- What constitutes a good null model?

**Preserve in- and out-degree**

# Generation of randomized networks

**Network randomization algorithm :**

- Start with the real network and repeatedly swap randomly chosen pairs of connections
  (X1→Y1, X2→Y2 is replaced by X1→Y2, X2→Y1)

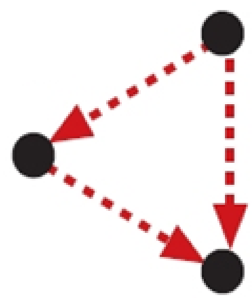



*(Switching is prohibited if the either of the X1→Y2 or X2→Y1 already exist)*

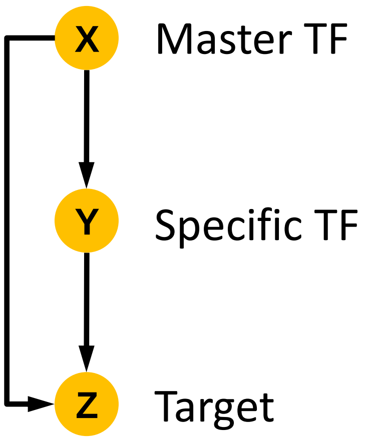
- Repeat until the network is “well randomized”

# Motifs in transcriptional regulatory networks

- E. Coli network
  - 424 operons (116 TFs)
  - 577 interactions
  - Significant enrichment of motif # 5



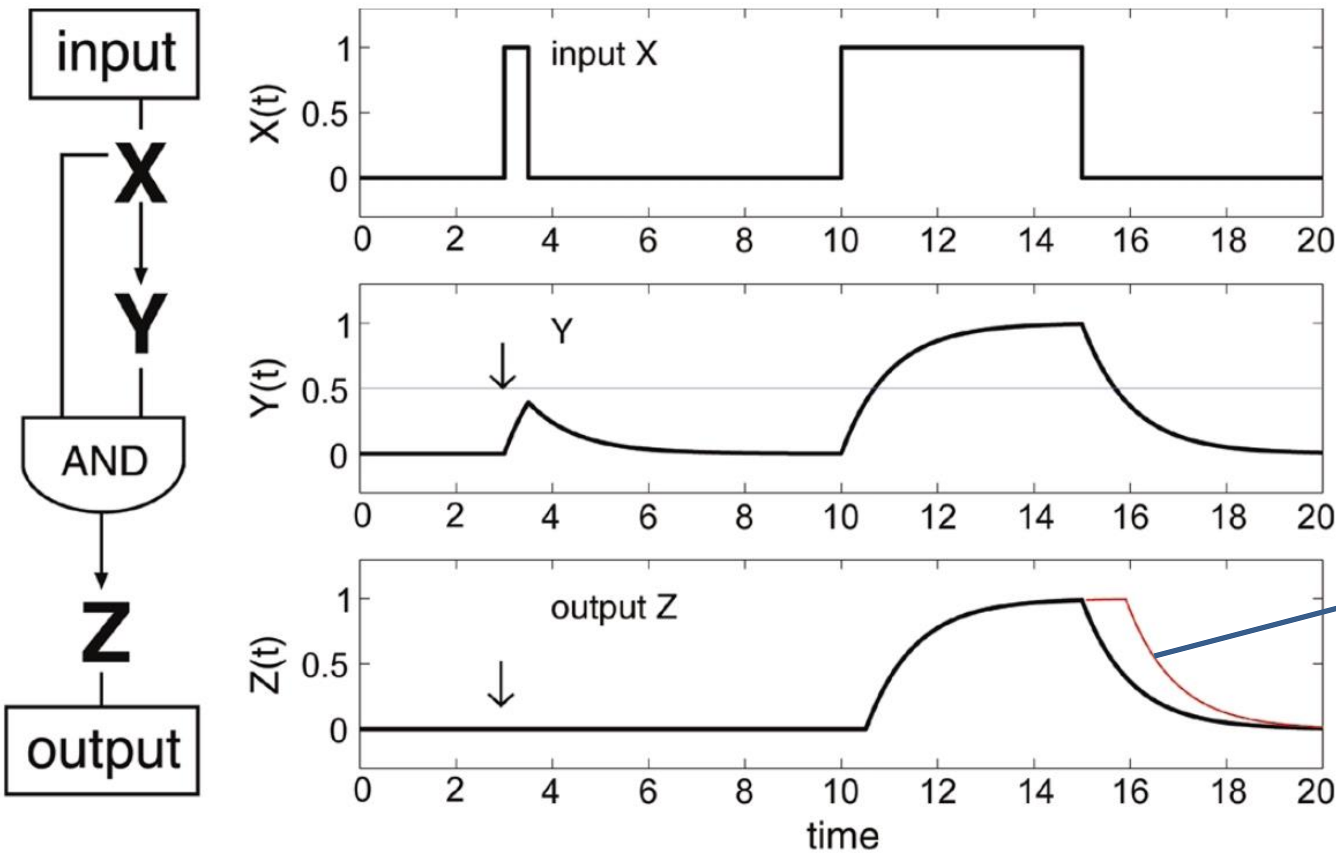
**(40 instances vs. 7±3)**




**Feed-Forward Loop (FFL)**

S. Shen-Orr et al. Nature Genetics 2002

# What’s so interesting about FFLs




Boolean Kinetics

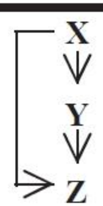
$$dY/dt = F(X, T_y) - aY$$

$$dZ/dt = F(X, T_y)F(Y, T_z) - aZ$$

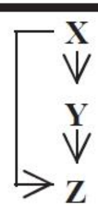
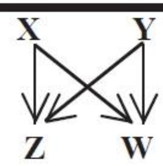
**A simple cascade has slower shutdown**

A coherent feed-forward loop can act as a circuit that rejects transient activation signals from the general transcription factor and responds only to persistent signals, while allowing for a rapid system shutdown.

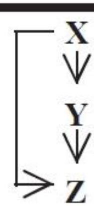
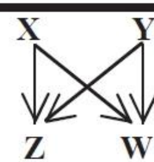
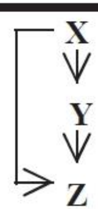
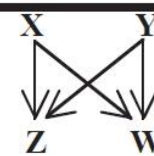
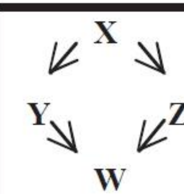
# Network motifs in biological networks

| Network | Nodes | Edges | $N_{real}$ | $N_{rand} \pm SD$ | $Z$ score |
|---|---|---|---|---|---|
| **Gene regulation (transcription)** | | | |  | **Feed-forward loop** |
| *E. coli* | 424 | 519 | 40 | 7 ± 3 | 10 |
| *S. cerevisiae** | 685 | 1,052 | 70 | 11 ± 4 | 14 |

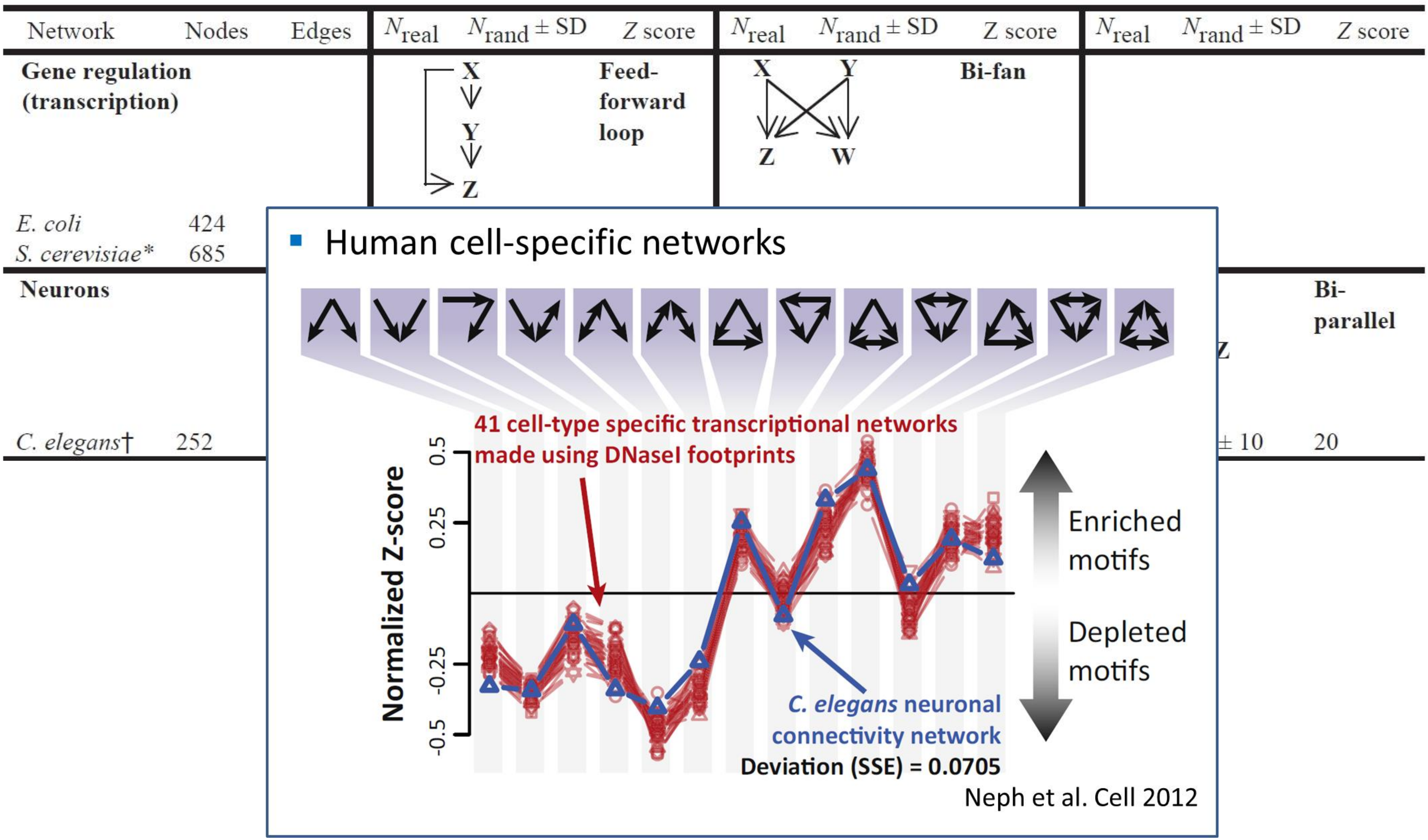
# Network motifs in biological networks

| Network | Nodes | Edges | $N_{real}$ | $N_{rand} \pm SD$ | $Z$ score | $N_{real}$ | $N_{rand} \pm SD$ | $Z$ score | $N_{real}$ | $N_{rand} \pm SD$ | $Z$ score |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Gene regulation (transcription)** | | |  | | **Feed-forward loop** |  | | **Bi-fan** | | | |
| *E. coli* | 424 | 519 | 40 | $7 \pm 3$ | 10 | 203 | $47 \pm 12$ | 13 | | | |
| *S. cerevisiae** | 685 | 1,052 | 70 | $11 \pm 4$ | 14 | 1812 | $300 \pm 40$ | 41 | | | |

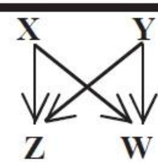
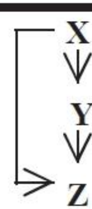
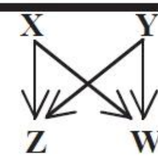
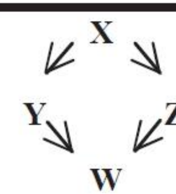
# Network motifs in biological networks

| Network | Nodes | Edges | $N_{real}$ | $N_{rand} \pm$ SD | $Z$ score | $N_{real}$ | $N_{rand} \pm$ SD | $Z$ score | $N_{real}$ | $N_{rand} \pm$ SD | $Z$ score |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Gene regulation (transcription)** | | |  | | **Feed-forward loop** |  | | **Bi-fan** | | | |
| *E. coli* | 424 | 519 | 40 | 7 ± 3 | 10 | 203 | 47 ± 12 | 13 | | | |
| *S. cerevisiae** | 685 | 1,052 | 70 | 11 ± 4 | 14 | 1812 | 300 ± 40 | 41 | | | |
| **Neurons** | | |  | | **Feed-forward loop** |  | | **Bi-fan** |  | | **Bi-parallel** |
| *C. elegans*† | 252 | 509 | 125 | 90 ± 10 | 3.7 | 127 | 55 ± 13 | 5.3 | 227 | 35 ± 10 | 20 |

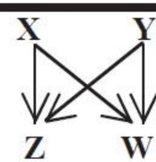
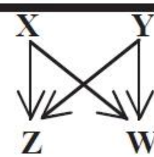
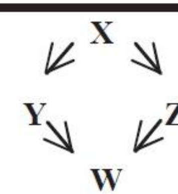
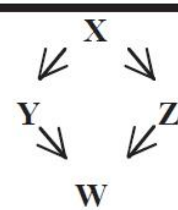
# Network motifs in biological networks



# Network motifs in biological networks

| Network | Nodes | Edges | $N_{real}$ | $N_{rand} \pm SD$ | $Z$ score | $N_{real}$ | $N_{rand} \pm SD$ | $Z$ score | $N_{real}$ | $N_{rand} \pm SD$ | $Z$ score |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Gene regulation (transcription)** | | |   | | **Feed-forward loop** |   | | **Bi-fan** | | | |
| *E. coli* | 424 | 519 | 40 | $7 \pm 3$ | 10 | 203 | $47 \pm 12$ | 13 | | | |
| *S. cerevisiae** | 685 | 1,052 | 70 | $11 \pm 4$ | 14 | 1812 | $300 \pm 40$ | 41 | | | |
| **Neurons** | | |   | | **Feed-forward loop** |   | | **Bi-fan** |   | | **Bi-parallel** |
| *C. elegans*† | 252 | 509 | 125 | $90 \pm 10$ | 3.7 | 127 | $55 \pm 13$ | 5.3 | 227 | $35 \pm 10$ | 20 |

# Network motifs in biological networks

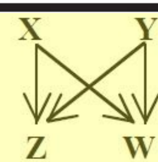
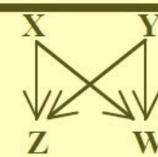
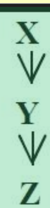
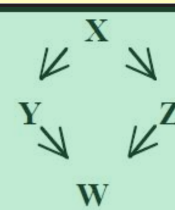
| Network | Nodes | Edges | $N_{real}$ | $N_{rand} \pm SD$ | $Z$ score | $N_{real}$ | $N_{rand} \pm SD$ | $Z$ score | $N_{real}$ | $N_{rand} \pm SD$ | $Z$ score |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Gene regulation (transcription)** | | | |  | **Feed-forward loop** | |  | **Bi-fan** | | | |
| *E. coli* | 424 | [illegible] | [illegible] | 7 ± 3 | 10 | 203 | 47 ± 12 | 13 | | | |
| *S. cerevisiae** | 685 | [illegible] | [illegible] | 11 ± 4 | 14 | 1812 | 300 ± 40 | 41 | | | |
| **Neurons** | | | |  | **Feed-forward loop** | |  | **Bi-fan** | |  | **Bi-parallel** |
| *C. elegans*† | 252 | 509 | 125 | 90 ± 10 | 3.7 | 127 | 55 ± 13 | 5.3 | 227 | 35 ± 10 | 20 |
| **Food webs** | | | |  | **Three chain** | |  | **Bi-parallel** | | | |
| Little Rock | 92 | 984 | 3219 | 3120 ± 50 | 2.1 | 7295 | 2220 ± 210 | 25 | | | |
| Ythan | 83 | 391 | 1182 | 1020 ± 20 | 7.2 | 1357 | 230 ± 50 | 23 | | | |
| St. Martin | 42 | 205 | 469 | 450 ± 10 | NS | 382 | 130 ± 20 | 12 | | | |
| Chesapeake | 31 | 67 | 80 | 82 ± 4 | NS | 26 | 5 ± 2 | 8 | | | |
| Coachella | 29 | 243 | 279 | 235 ± 12 | 3.6 | 181 | 80 ± 20 | 5 | | | |
| Skipwith | 25 | 189 | 184 | 150 ± 7 | 5.5 | 397 | 80 ± 25 | 13 | | | |
| B. Brook | 25 | 104 | 181 | 130 ± 7 | 7.4 | 267 | 30 ± 7 | 32 | | | |

**Why do these networks have similar motifs?**

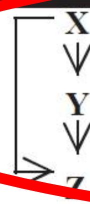
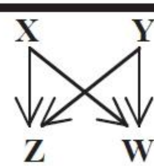
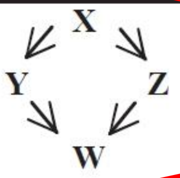
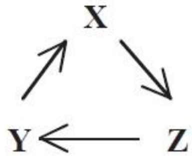
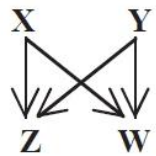
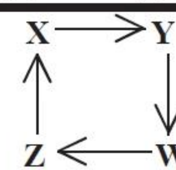
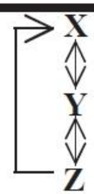
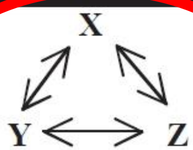
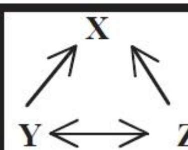
**Why is this network so different?**

**FFL motif is under-represented!**

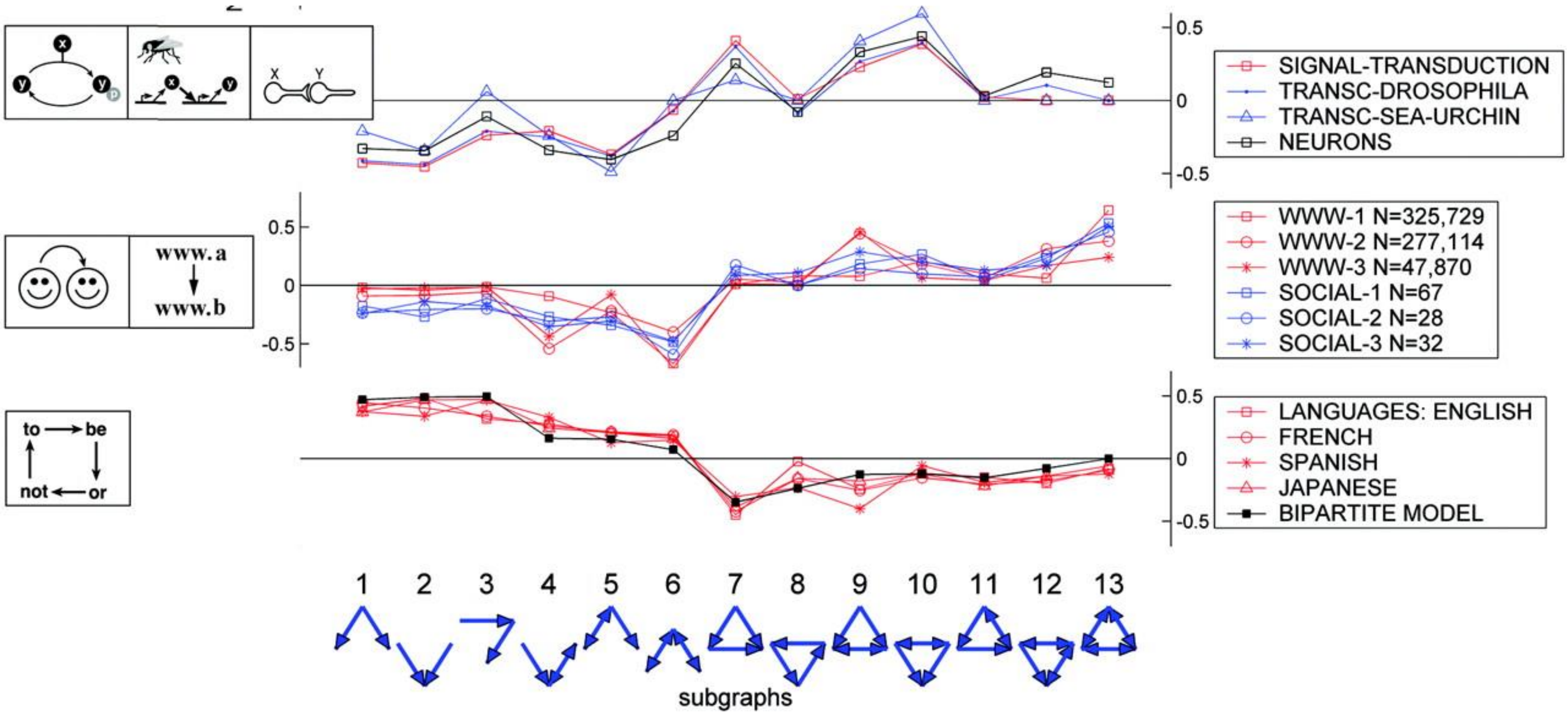
# Information Flow vs. Energy Flow

| Network | Nodes | Edges | $N_{real}$ | $N_{rand} \pm SD$ | Z score | $N_{real}$ | $N_{rand} \pm SD$ | Z score | $N_{real}$ | $N_{rand} \pm SD$ | Z score |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Gene regulation (transcription)** | | | |  | **Feed-forward loop** | |  | **Bi-fan** | | | |
| *E. coli* | 424 | 519 | 40 | 7 ± 3 | 10 | 203 | 47 ± 12 | 13 | | | |
| *S. cerevisiae** | 685 | 1,052 | 70 | 11 ± 4 | 14 | 1812 | 300 ± 40 | 41 | | | |
| **Neurons** | | | |  | **Feed-forward loop** | |  | **Bi-fan** | |  | **Bi-parallel** |
| *C. elegans*† | 252 | 509 | 125 | 90 ± 10 | 3.7 | 127 | 55 ± 13 | 5.3 | 227 | 35 ± 10 | 20 |
| **Food webs** | | | |  | **Three chain** | |  | **Bi-parallel** | **FFL motif is under-represented!** | | |
| Little Rock | 92 | 984 | 3219 | 3120 ± 50 | 2.1 | 7295 | 2220 ± 210 | 25 | | | |
| Ythan | 83 | 391 | 1182 | 1020 ± 20 | 7.2 | 1357 | 230 ± 50 | 23 | | | |
| St. Martin | 42 | 205 | 469 | 450 ± 10 | NS | 382 | 130 ± 20 | 12 | | | |
| Chesapeake | 31 | 67 | 80 | 82 ± 4 | NS | 26 | 5 ± 2 | 8 | | | |
| Coachella | 29 | 243 | 279 | 235 ± 12 | 3.6 | 181 | 80 ± 20 | 5 | | | |
| Skipwith | 25 | 189 | 184 | 150 ± 7 | 5.5 | 397 | 80 ± 25 | 13 | | | |
| B. Brook | 25 | 104 | 181 | 130 ± 7 | 7.4 | 267 | 30 ± 7 | 32 | | | |

# Network Motifs in Technological Networks

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Electronic circuits (forward logic chips)** | | |  | | **Feed-forward loop** |  | | **Bi-fan** |  | | **Bi-parallel** |
| s15850 | 10,383 | 14,240 | 424 | 2 ± 2 | 285 | 1040 | 1 ± 1 | 1200 | 480 | 2 ± 1 | 335 |
| s38584 | 20,717 | 34,204 | 413 | 10 ± 3 | 120 | 1739 | 6 ± 2 | 800 | 711 | 9 ± 2 | 320 |
| s38417 | 23,843 | 33,661 | 612 | 3 ± 2 | 400 | 2404 | 1 ± 1 | 2550 | 531 | 2 ± 2 | 340 |
| s9234 | 5,844 | 8,197 | 211 | 2 ± 1 | 140 | 754 | 1 ± 1 | 1050 | 209 | 1 ± 1 | 200 |
| s13207 | 8,651 | 11,831 | 403 | 2 ± 1 | 225 | 4445 | 1 ± 1 | 4950 | 264 | 2 ± 1 | 200 |
| **Electronic circuits (digital fractional multipliers)** | | |  | | **Three-node feedback loop** |  | | **Bi-fan** |  | | **Four-node feedback loop** |
| s208 | 122 | 189 | 10 | 1 ± 1 | 9 | 4 | 1 ± 1 | 3.8 | 5 | 1 ± 1 | 5 |
| s420 | 252 | 399 | 20 | 1 ± 1 | 18 | 10 | 1 ± 1 | 10 | 11 | 1 ± 1 | 11 |
| s838‡ | 512 | 819 | 40 | 1 ± 1 | 38 | 22 | 1 ± 1 | 20 | 23 | 1 ± 1 | 25 |
| **World Wide Web** | | |  | | **Feedback with two mutual dyads** |  | | **Fully connected triad** |  | | **Uplinked mutual dyad** |
| nd.edu§ | 325,729 | 1.46e6 | 1.1e5 | 2e3 ± 1e2 | 800 | 6.8e6 | 5e4±4e2 | 15,000 | 1.2e6 | 1e4 ± 2e2 | 5000 |

# Motif-based network super-families




R. Milo et al. Superfamilies of evolved and designed networks. Science, 2004