

Biological Networks Analysis

Introduction and Dijkstra's algorithm

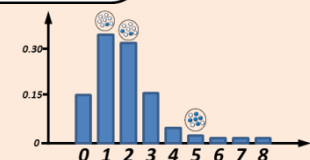
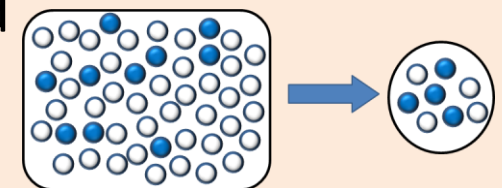
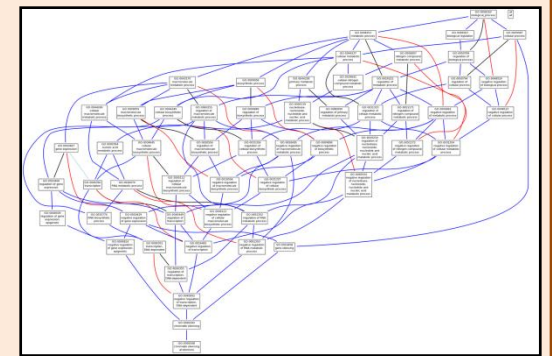
Genome 373

Genomic Informatics

Elhanan Borenstein

A quick review

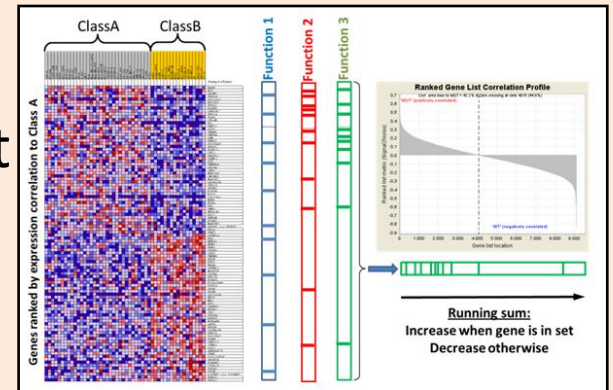
- Gene expression profiling
 - Which molecular processes/functions are involved in a certain phenotype (e.g., disease, stress response, etc.)
- The Gene Ontology (GO) Project
 - Provides shared vocabulary/annotation
 - Terms are linked in a complex structure
- Enrichment analysis:
 - Find the “most” differentially expressed genes
 - Identify **over-represented** annotations
 - Modified Fisher's exact test



A quick review – cont'

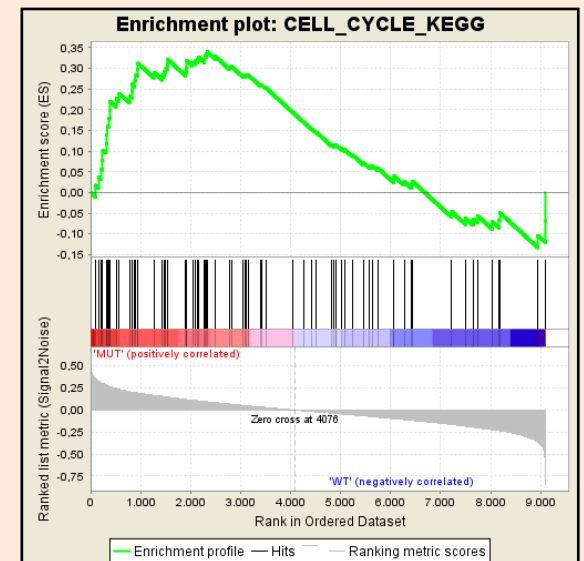
■ Gene Set Enrichment Analysis

- Calculates a score for the enrichment of a entire set of genes
- Does not require setting a cutoff!
- Identifies the set of relevant genes!
- Provides a more robust statistical framework!



■ GSEA steps:

1. Calculation of an enrichment score (ES) for each functional category
2. Estimation of significance level
3. Adjustment for multiple hypotheses testing



Biological networks

What is a network?

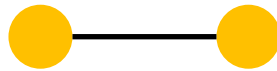
What networks are used in biology?

Why do we need networks (and network theory)?

How do we find the shortest path between two nodes?

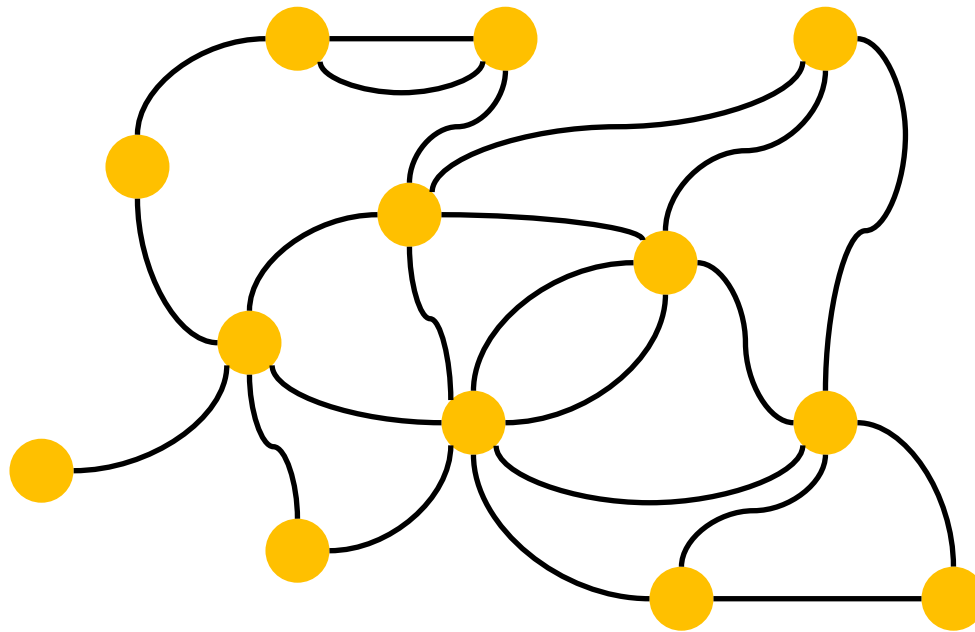
What is a network?

- A map of interactions or relationships
- A collection of **nodes** and **links (edges)**



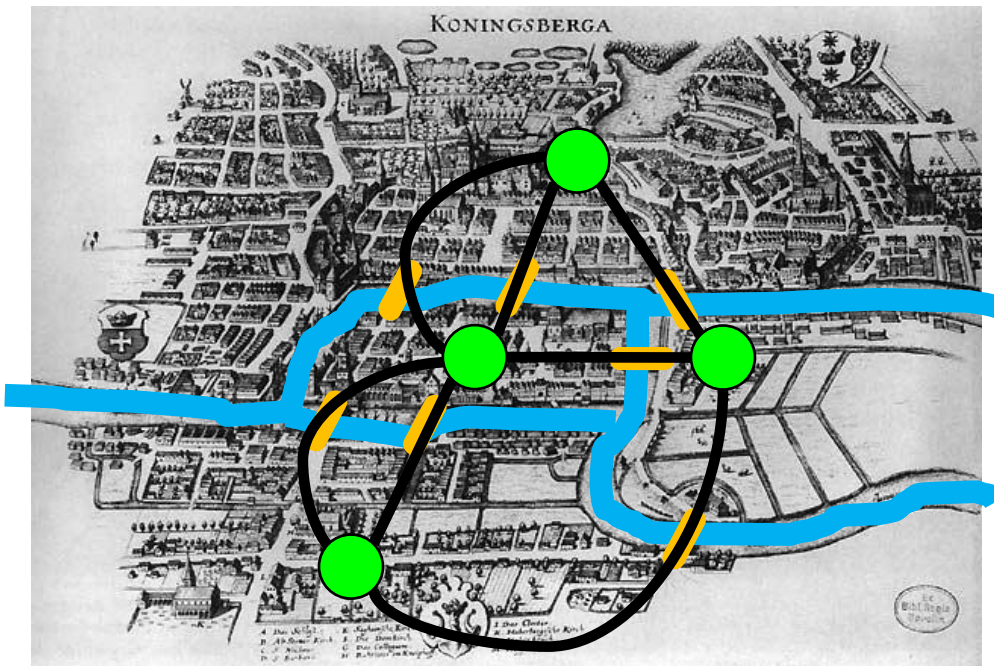
What is a network?

- A map of interactions or relationships
- A collection of **nodes** and **links (edges)**



Networks as Tools

- **The Seven Bridges of Königsberg**
 - Published by **Leonhard Euler**, 1736
 - Considered the first paper in graph theory

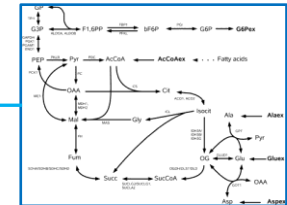
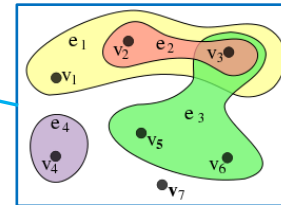
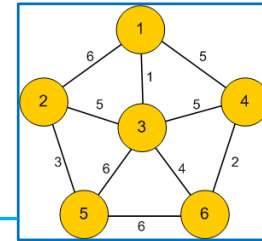
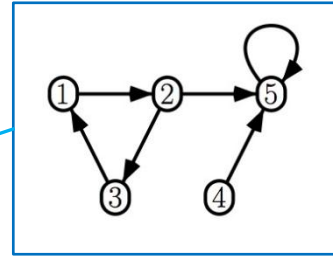


Leonhard Euler
1707 –1783

Types of networks

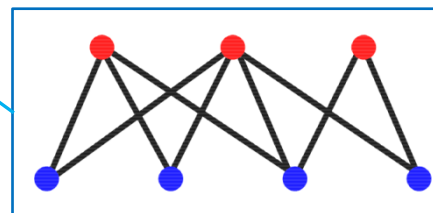
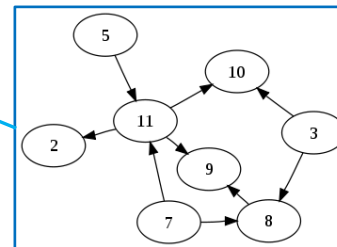
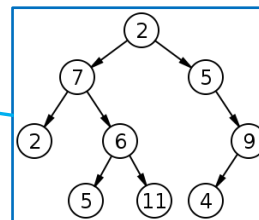
Edges:

- Directed/undirected
- Weighted/non-weighted
- Simple-edges/Hyperedges



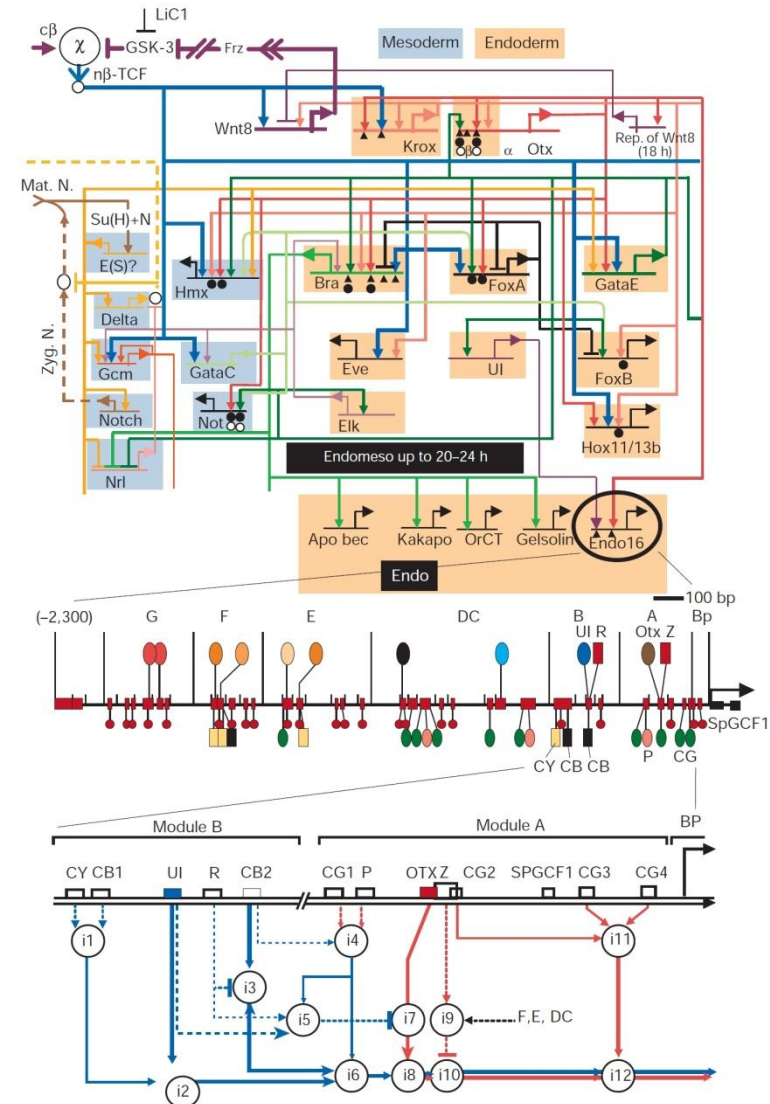
Special topologies:

- Directed Acyclic Graphs (DAG)
- Trees
- Bipartite networks



Transcriptional regulatory networks

- Reflect the cell's genetic regulatory circuitry
 - **Nodes:** transcription factors and genes;
 - **Edges:** from TF to the genes it regulates
 - Directed; weighted?; “almost” bipartite
- Derived through:
 - Chromatin IP
 - Microarrays
 - **Computationally**



Metabolic networks

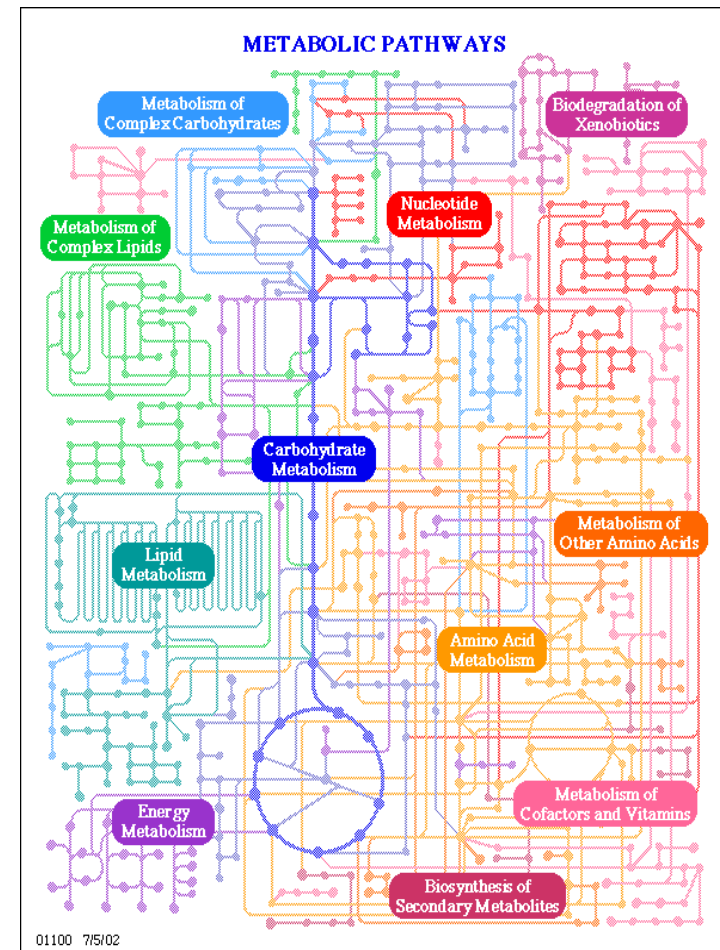
- Reflect the set of biochemical reactions in a cell
 - **Nodes:** metabolites
 - **Edges:** biochemical reactions
 - Directed; weighted?; hyperedges?
- Derived through:
 - Knowledge of biochemistry
 - Metabolic flux measurements
 - Homology?



S. Cerevisiae

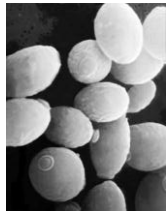
1062 metabolites

1149 reactions



Protein-protein interaction (PPI) networks

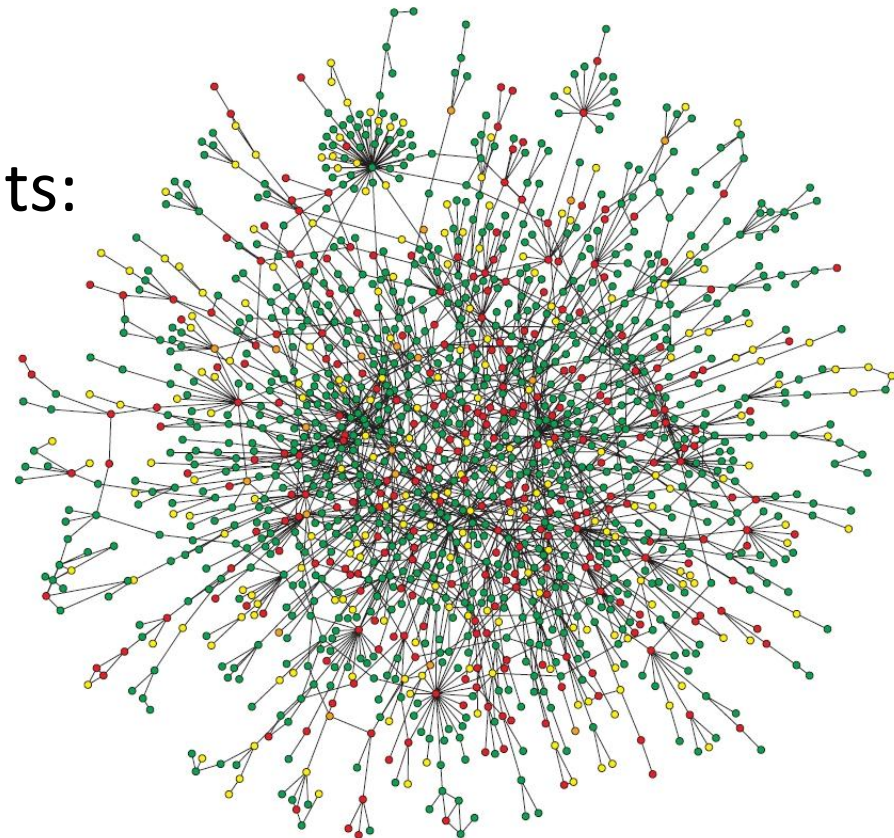
- Reflect the cell's molecular interactions and signaling pathways (interactome)
 - **Nodes:** proteins
 - **Edges:** interactions(?)
 - Undirected
- High-throughput experiments:
 - Protein Complex-IP (Co-IP)
 - Yeast two-hybrid
 - Computationally



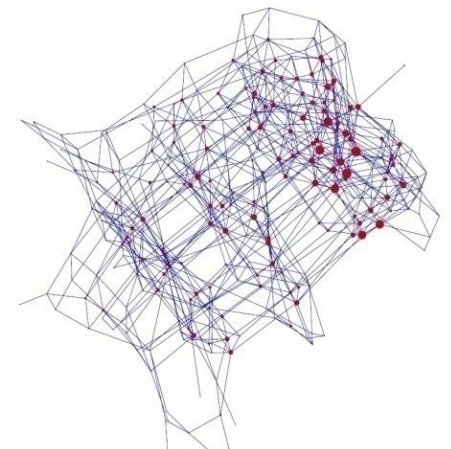
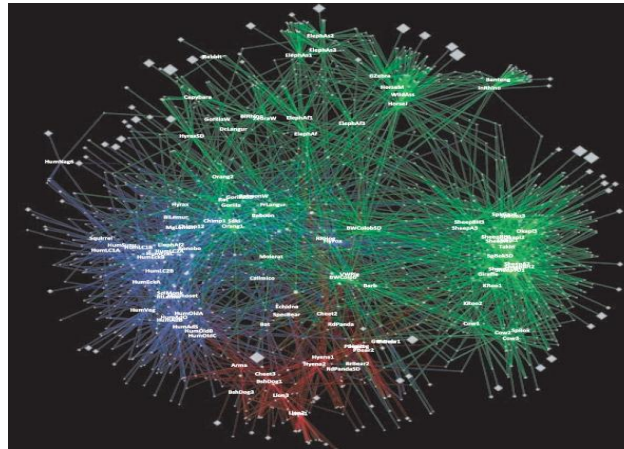
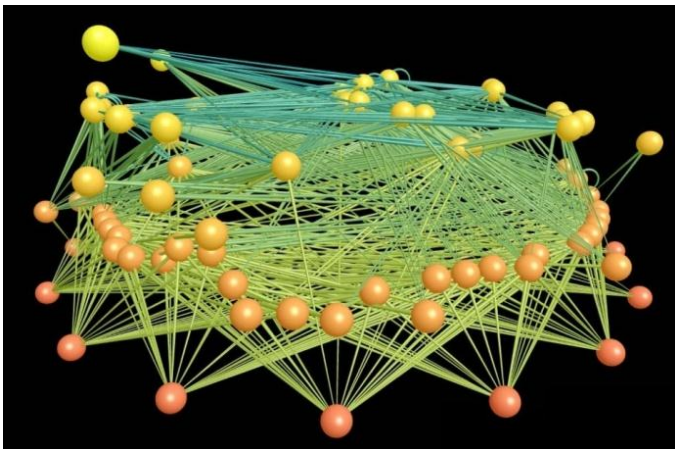
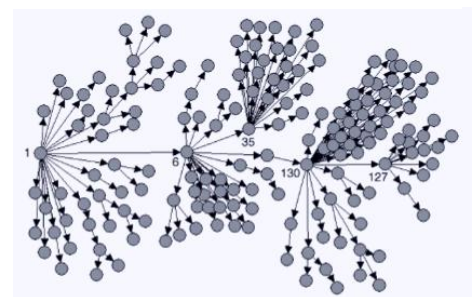
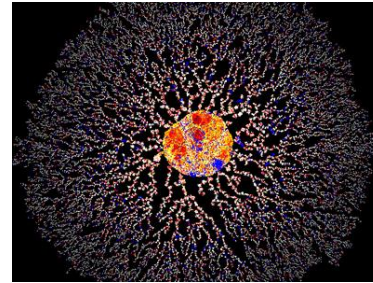
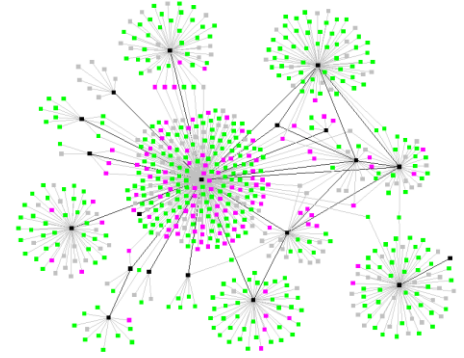
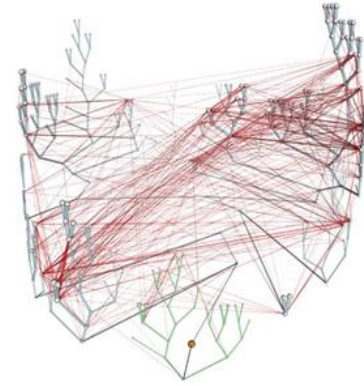
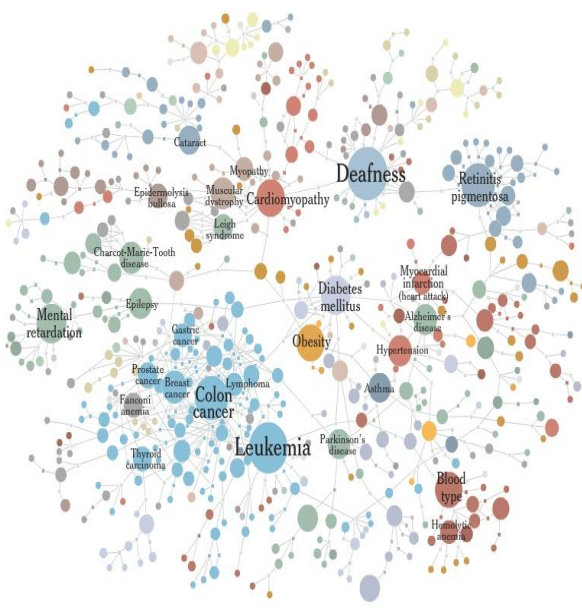
S. Cerevisiae

4389 proteins

14319 interactions



Other networks in biology/medicine



Non-biological networks

- **Computer related networks:**

- WWW; Internet backbone
- Communications and IP

- **Social networks:**

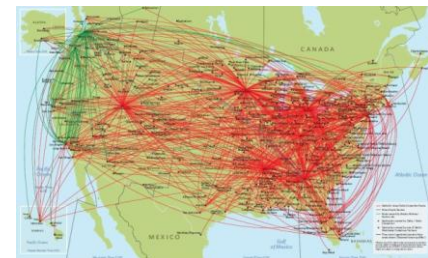
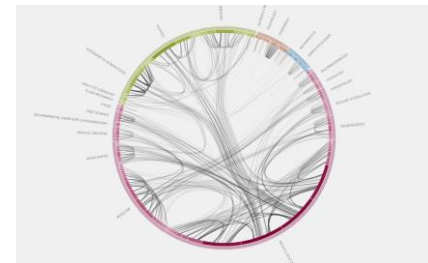
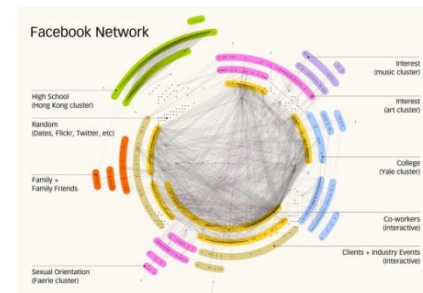
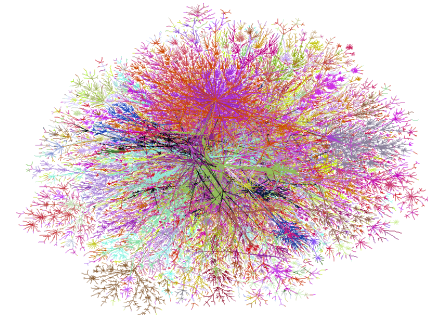
- Friendship (facebook; clubs)
- Citations / information flow
- Co-authorships (papers)
- Co-occurrence (movies; Jazz)

- **Transportation:**

- Highway systems; Airline routes

- **Electronic/Logic circuits**

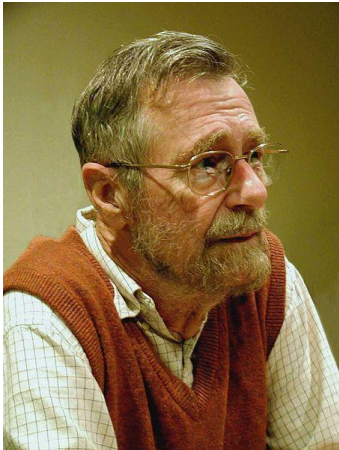
- Many many more...



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, Erdős 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:**
 - Iterative
 - 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:

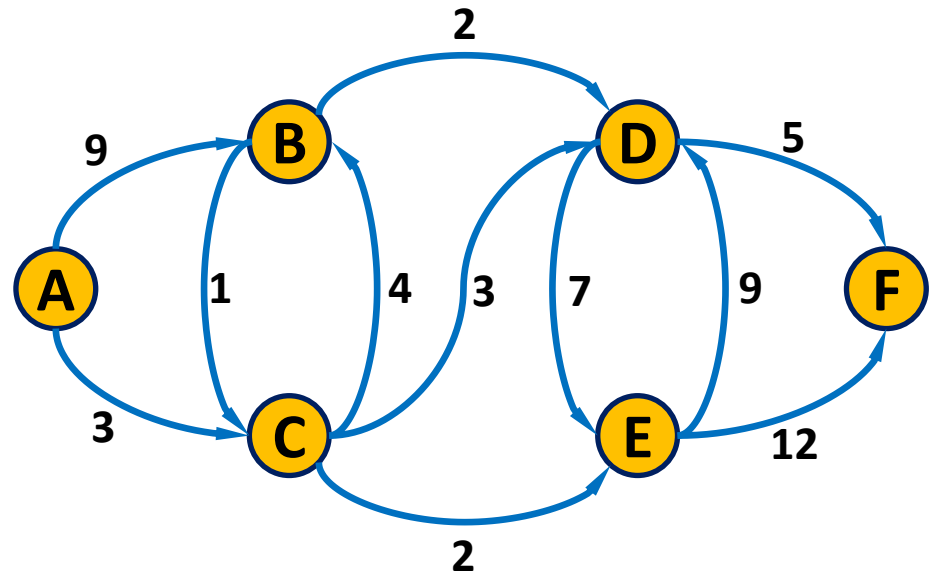
- i. Calculate tentative distance, D^t , through current node.
- ii. If D^t smaller than D (previously recorded distance): $D \leftarrow D^t$
- iii. Mark current node as visited (note: shortest dist. found).

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

- A simple synthetic network



1. Initialize:

- Assign a distance value, D , to each node.
Set D to zero for *start* node and to infinity for all others.
- Mark all nodes as unvisited.
- Set *start* node as current node.

2. For each of the current node's unvisited neighbors:

- Calculate tentative distance, D^t , through current node.
- If D^t smaller than D (previously recorded distance): $D \leftarrow D^t$
- Mark current node as visited (note: shortest dist. found).

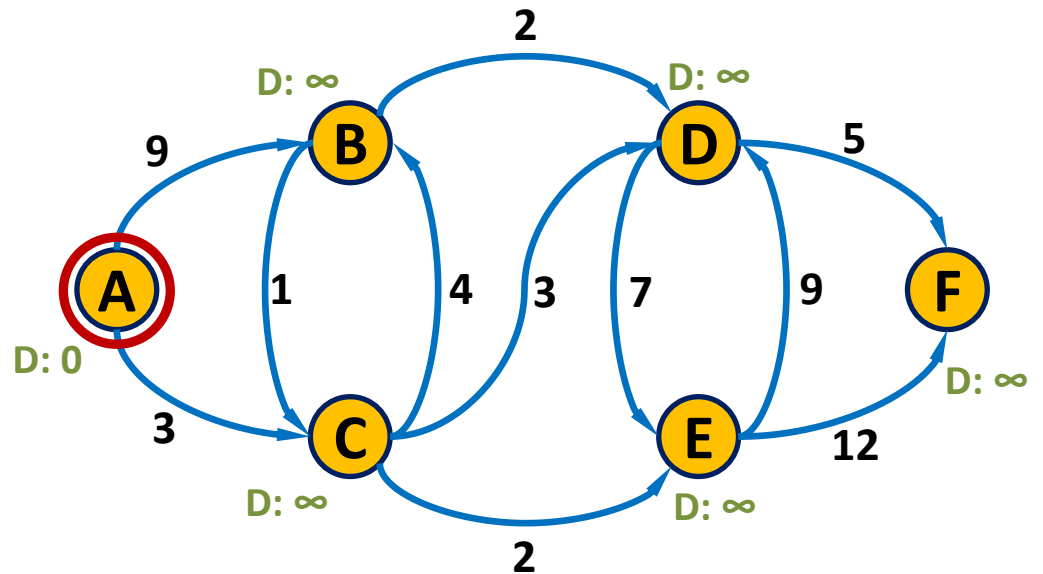
3. Set the unvisited node with the smallest distance as the next "current node" and continue from step 2.

4. Once all nodes are marked as visited, finish.

Dijkstra's algorithm

- Initialization
- Mark A (start) as current node

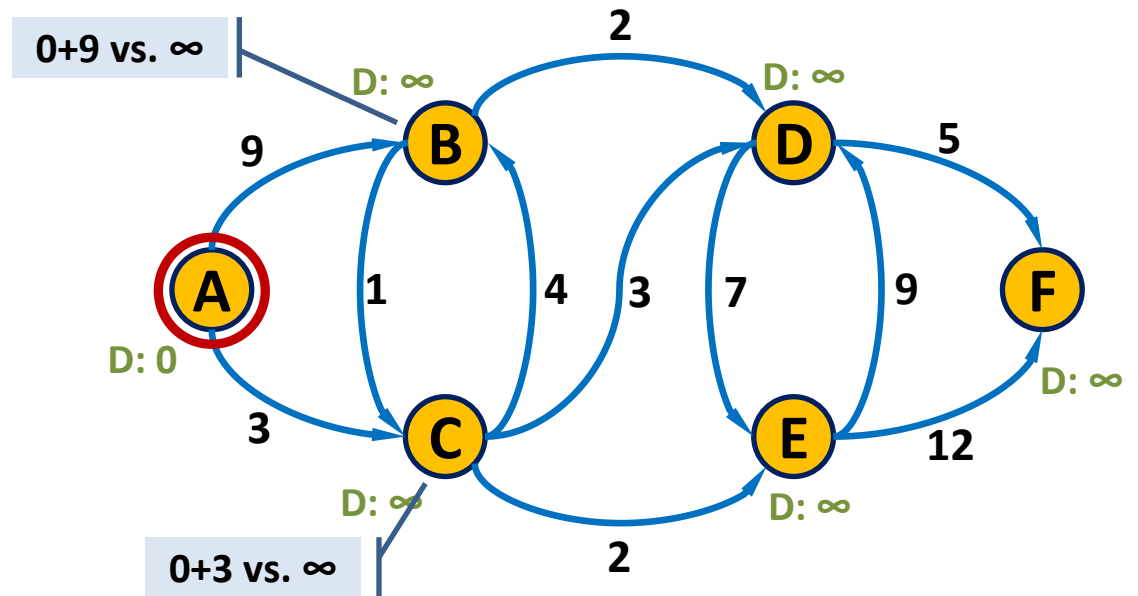
A	B	C	D	E	F
0	∞	∞	∞	∞	∞



Dijkstra's algorithm

- Check unvisited neighbors of A

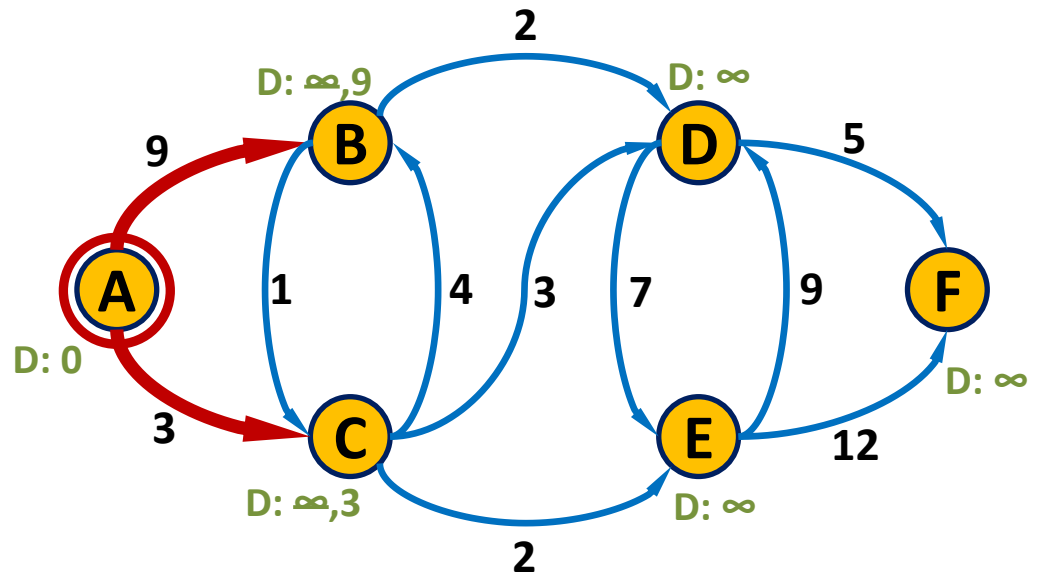
A	B	C	D	E	F
0	∞	∞	∞	∞	∞



Dijkstra's algorithm

- Update D
- Record path

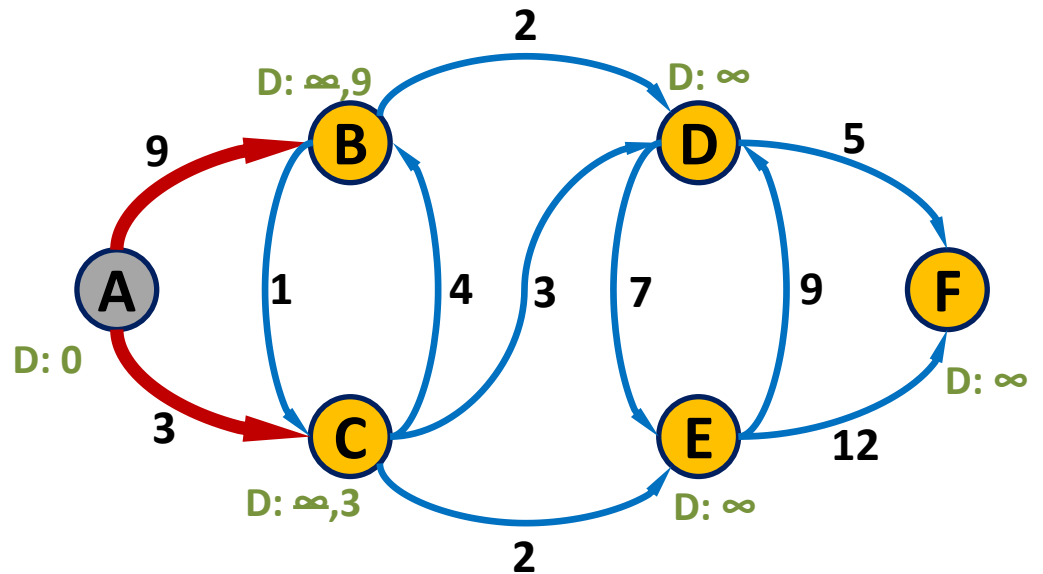
A	B	C	D	E	F
0	∞	∞	∞	∞	∞
0	9	3	∞	∞	∞



Dijkstra's algorithm

- Mark A as visited ...

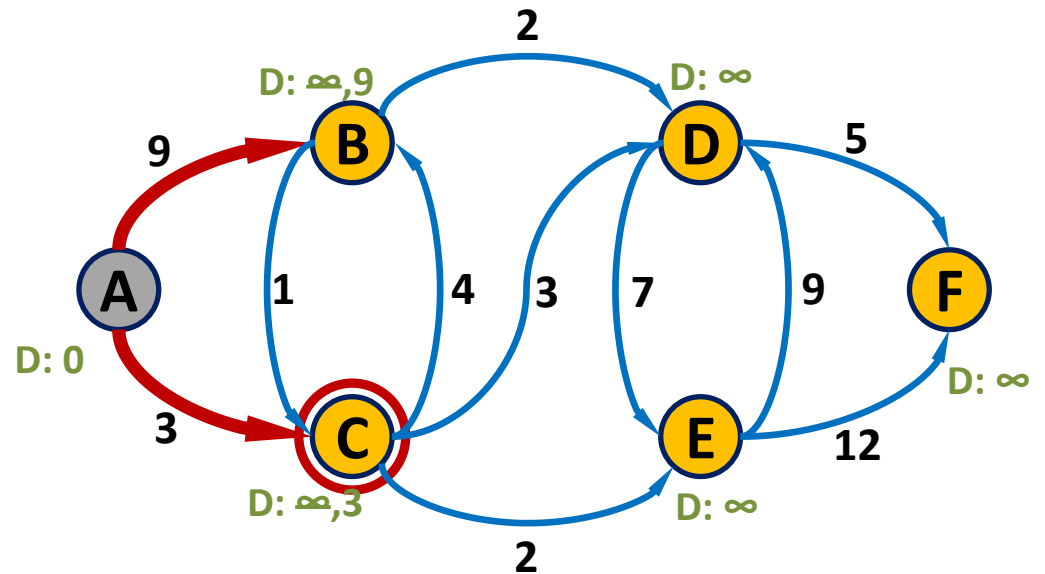
A	B	C	D	E	F
0	∞	∞	∞	∞	∞
0	9	3	∞	∞	∞



Dijkstra's algorithm

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

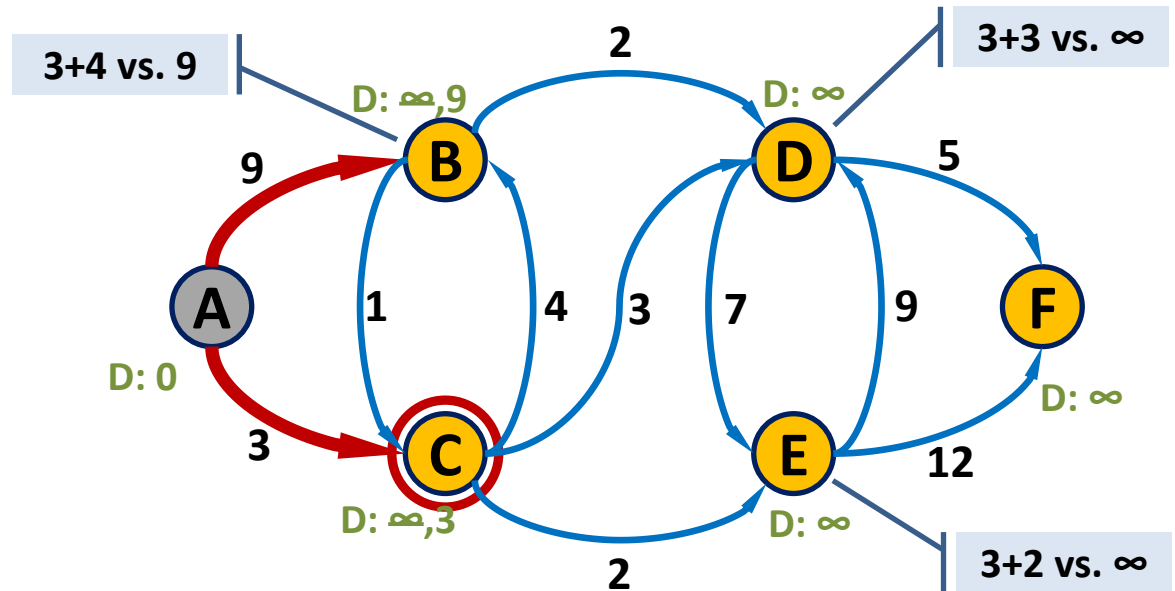
A	B	C	D	E	F
0	∞	∞	∞	∞	∞
0	9	3	∞	∞	∞



Dijkstra's algorithm

- Check unvisited neighbors of C

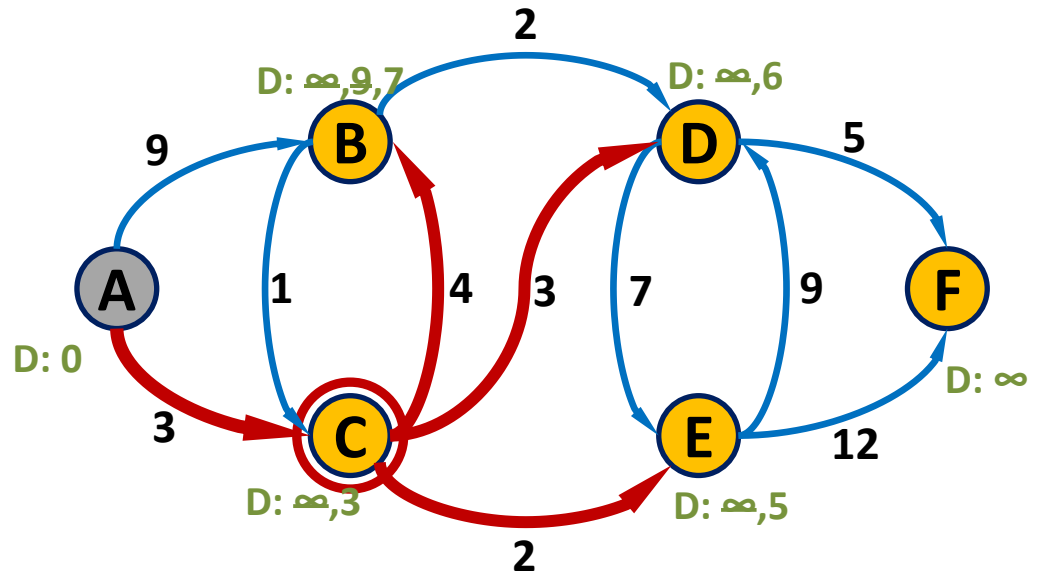
A	B	C	D	E	F
0	∞	∞	∞	∞	∞
0	9	3	∞	∞	∞



Dijkstra's algorithm

- Update distance
- Record path

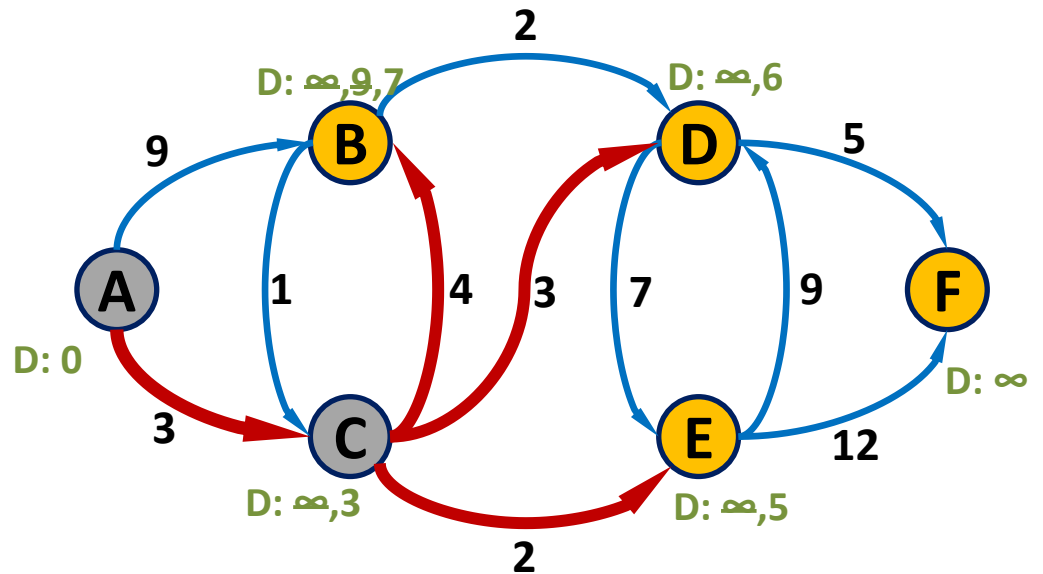
A	B	C	D	E	F
0	∞	∞	∞	∞	∞
0	9	3	∞	∞	∞
	7	3	6	5	∞



Dijkstra's algorithm

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

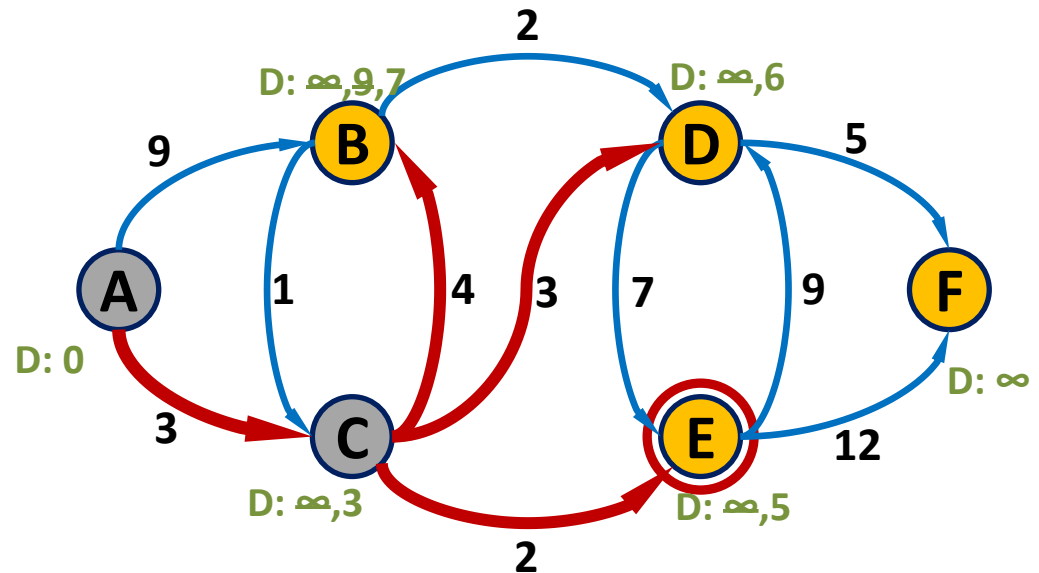
A	B	C	D	E	F
0	∞	∞	∞	∞	∞
0	9	3	∞	∞	∞
	7	3	6	5	∞



Dijkstra's algorithm

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

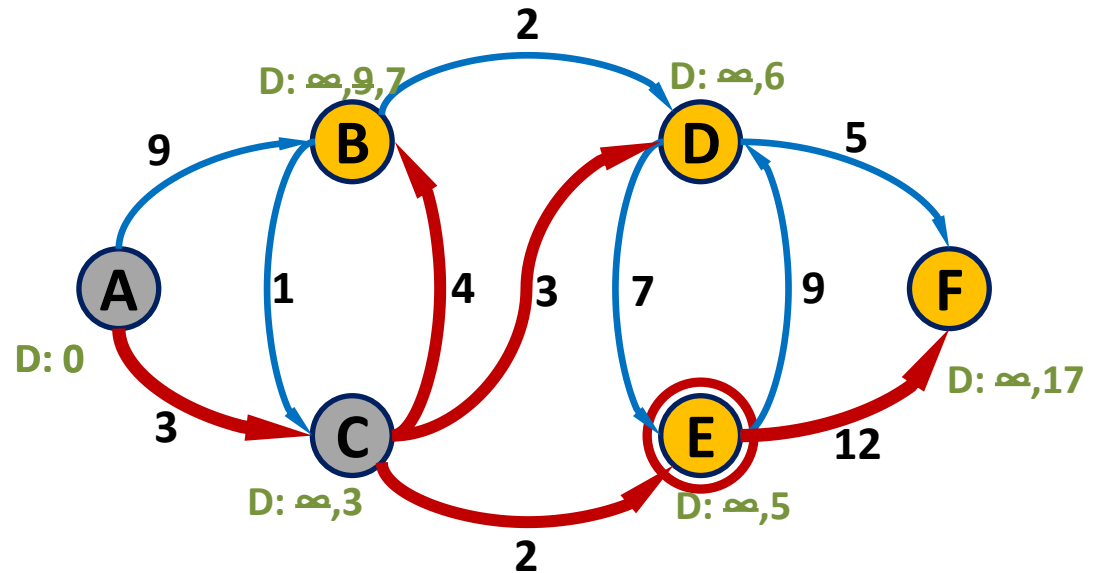
A	B	C	D	E	F
0	∞	∞	∞	∞	∞
0	9	3	∞	∞	∞
	7	3	6	5	∞



Dijkstra's algorithm

- Update D
- Record path

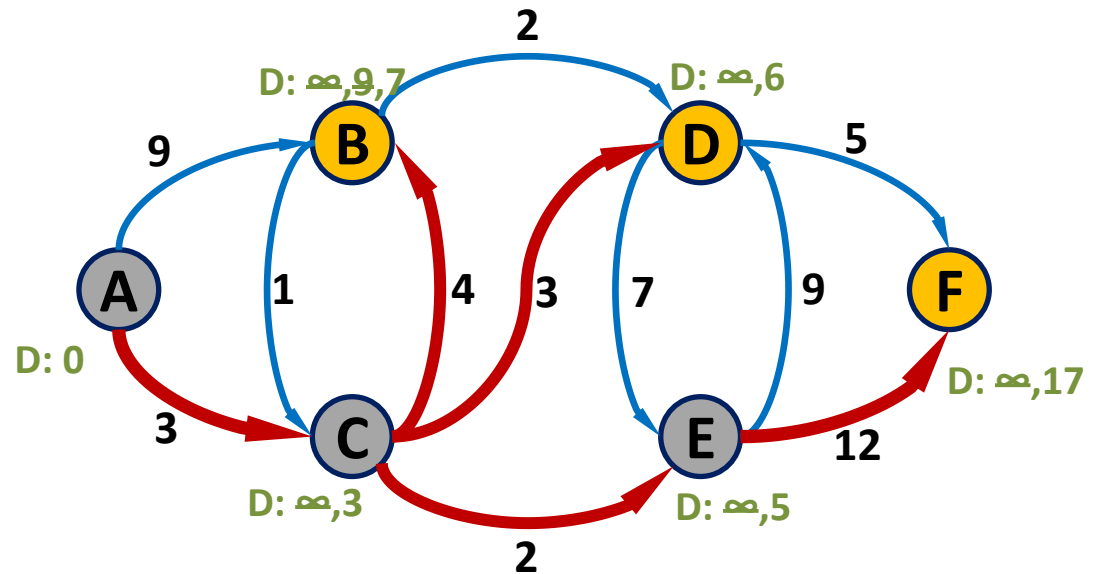
A	B	C	D	E	F
0	∞	∞	∞	∞	∞
0	9	3	∞	∞	∞
	7	3	6	5	∞
	7		6	5	17



Dijkstra's algorithm

- Mark E as visited

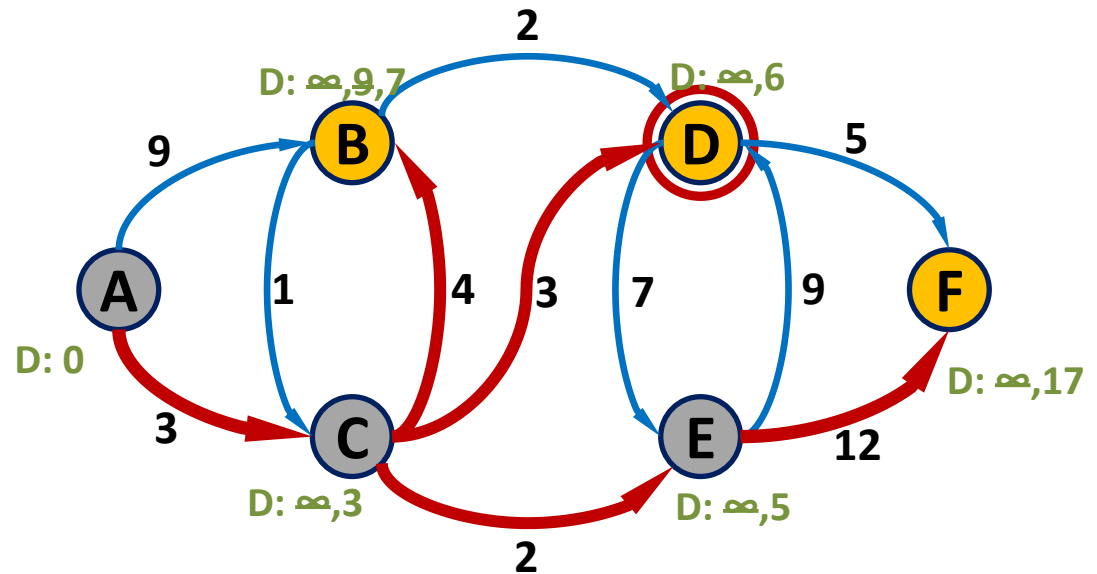
A	B	C	D	E	F
0	∞	∞	∞	∞	∞
0	9	3	∞	∞	∞
	7	3	6	5	∞
	7		6	5	17



Dijkstra's algorithm

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

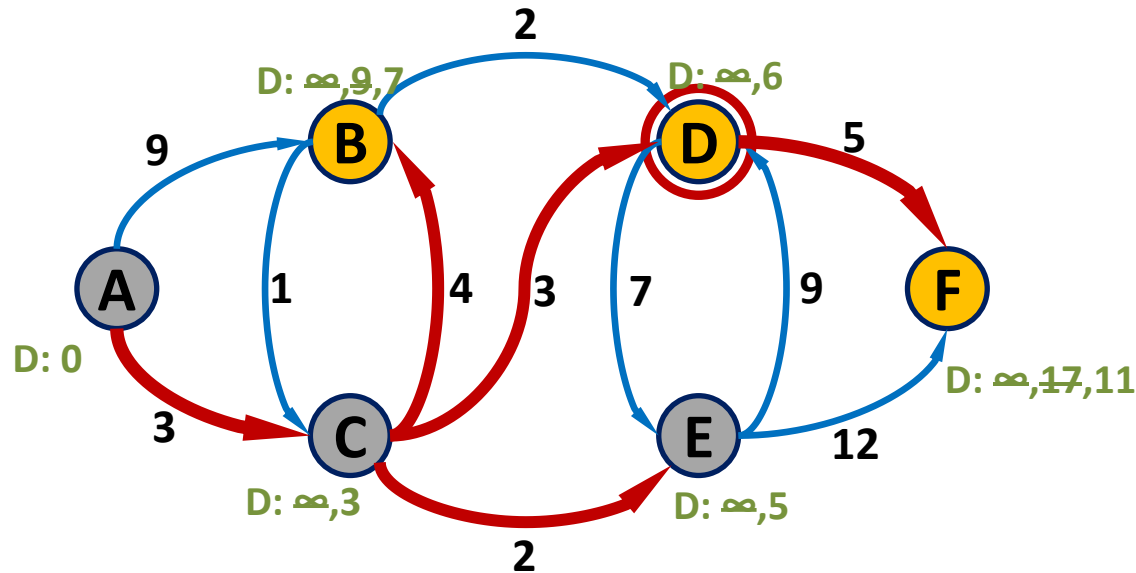
A	B	C	D	E	F
0	∞	∞	∞	∞	∞
0	9	3	∞	∞	∞
	7	3	6	5	∞
	7		6	5	17



Dijkstra's algorithm

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

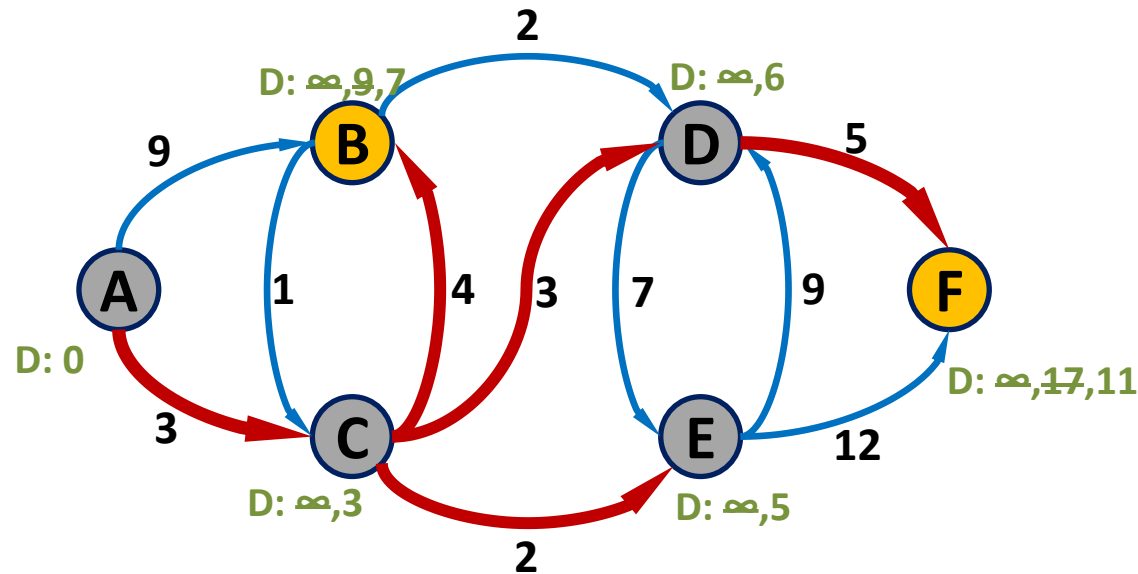
A	B	C	D	E	F
0	∞	∞	∞	∞	∞
0	9	3	∞	∞	∞
	7	3	6	5	∞
	7		6	5	17
	7		6		11



Dijkstra's algorithm

- Mark D as visited

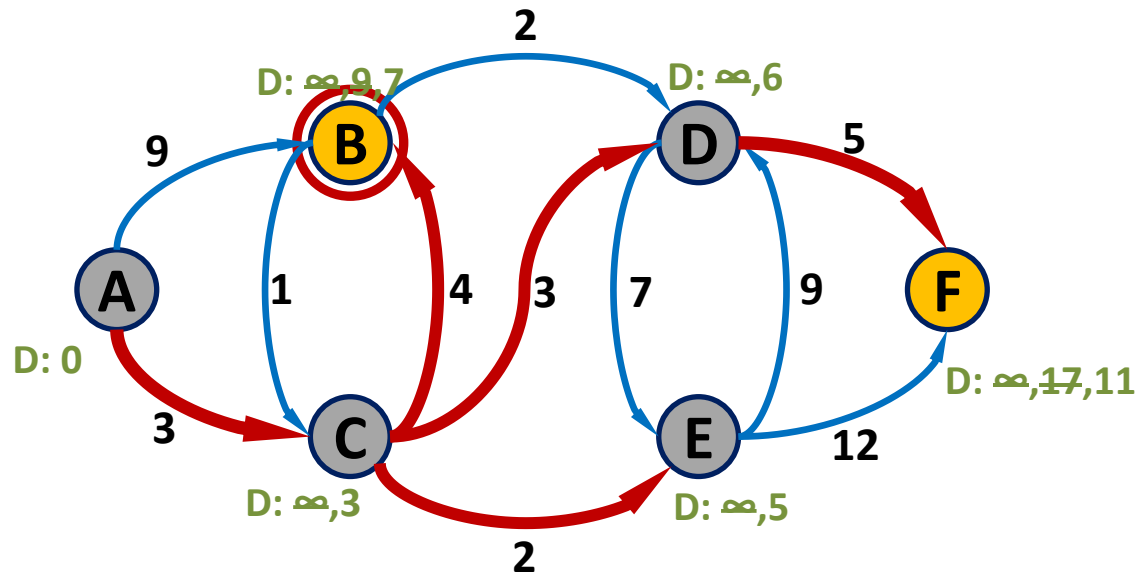
A	B	C	D	E	F
0	∞	∞	∞	∞	∞
0	9	3	∞	∞	∞
	7	3	6	5	∞
	7		6	5	17
	7		6		11



Dijkstra's algorithm

- Mark B as current node
- Check neighbors

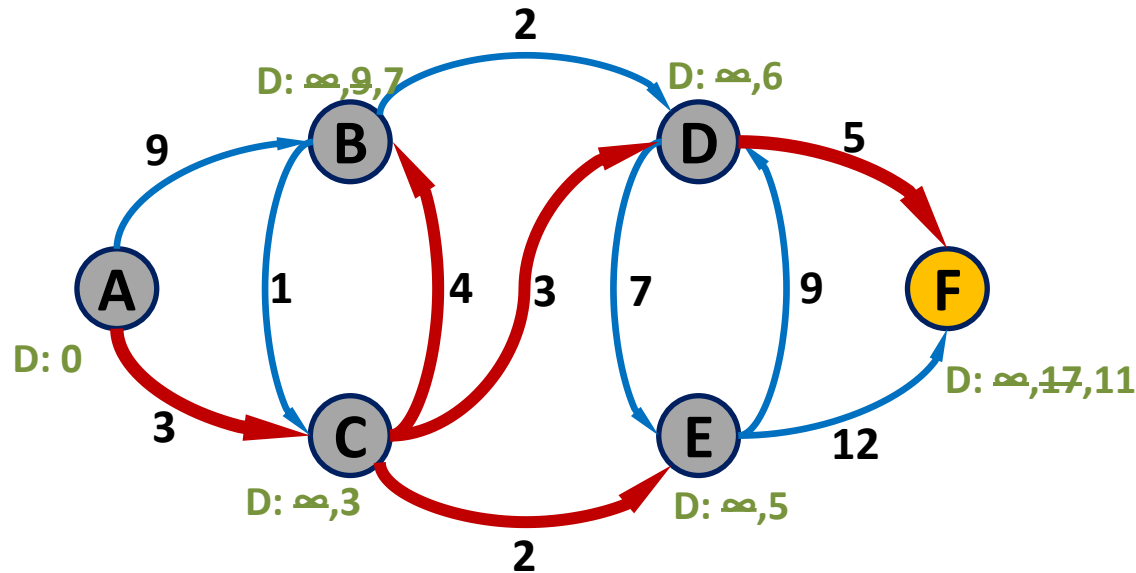
A	B	C	D	E	F
0	∞	∞	∞	∞	∞
0	9	3	∞	∞	∞
7	3	6	5	∞	
7		6	5	17	
7		6		11	



Dijkstra's algorithm

- No updates..
- Mark B as visited

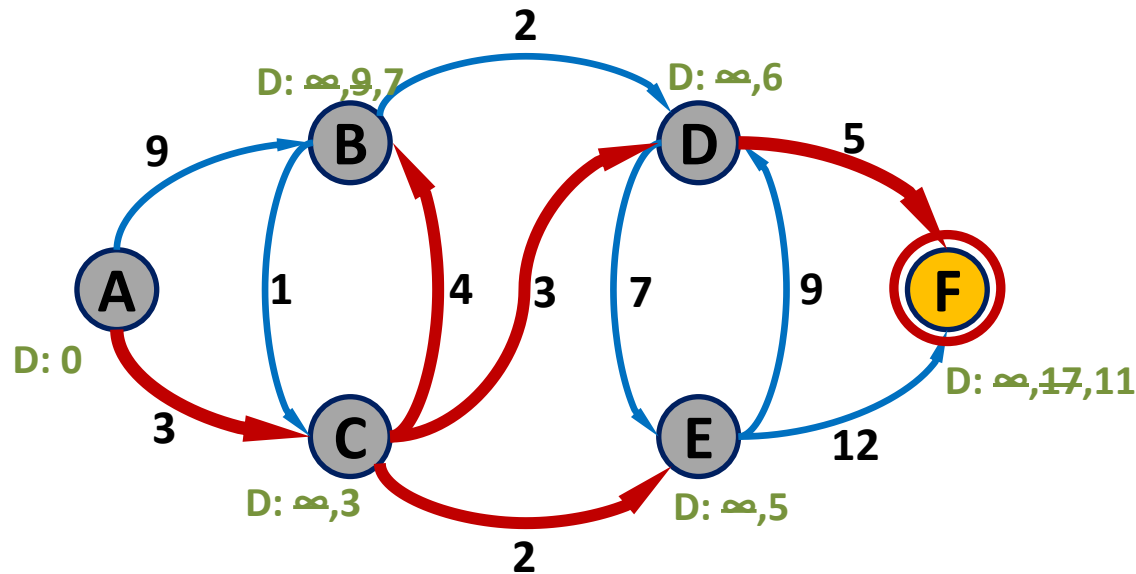
A	B	C	D	E	F
0	∞	∞	∞	∞	∞
0	9	3	∞	∞	∞
	7	3	6	5	∞
	7		6	5	17
	7		6		11
	7				11



Dijkstra's algorithm

- Mark F as current

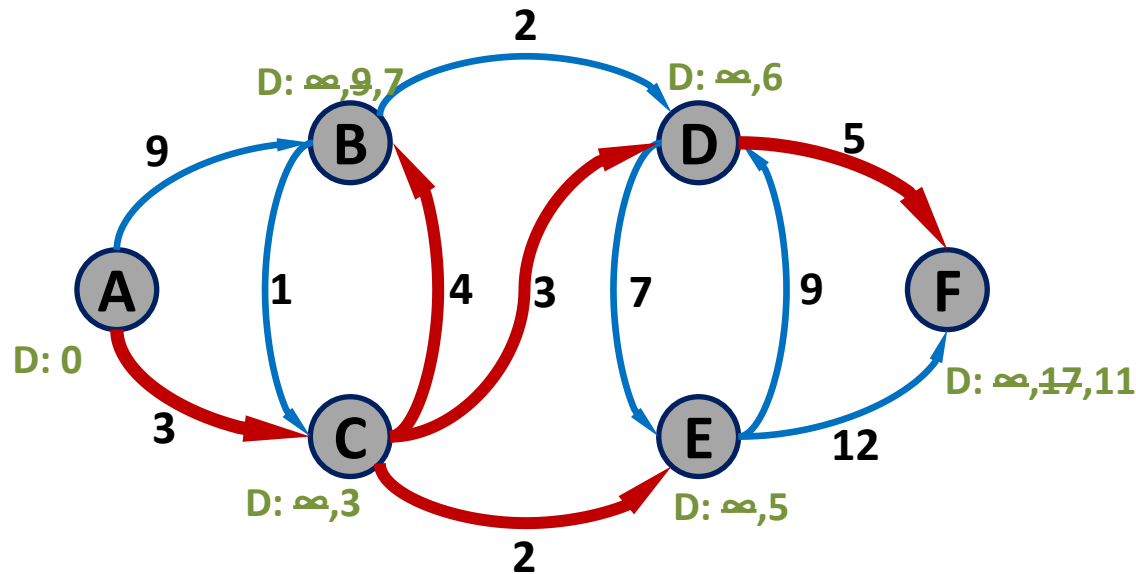
A	B	C	D	E	F
0	∞	∞	∞	∞	∞
0	9	3	∞	∞	∞
	7	3	6	5	∞
	7		6	5	17
	7		6		11
	7				11



Dijkstra's algorithm

- Mark F as visited

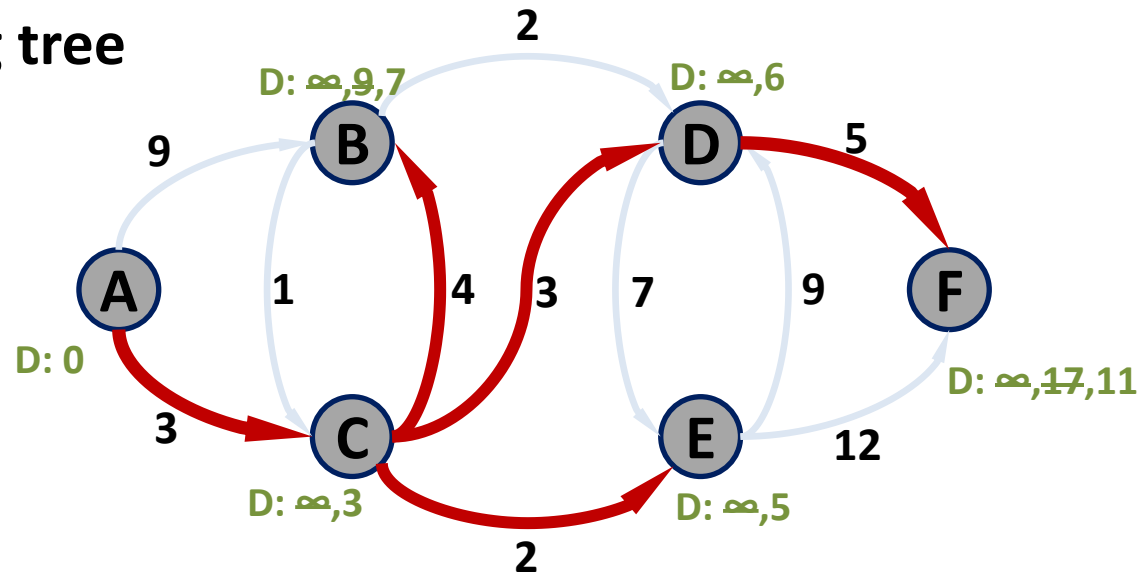
A	B	C	D	E	F
0	∞	∞	∞	∞	∞
0	9	3	∞	∞	∞
	7	3	6	5	∞
	7		6	5	17
	7		6		11
	7				11
					11



We are done!

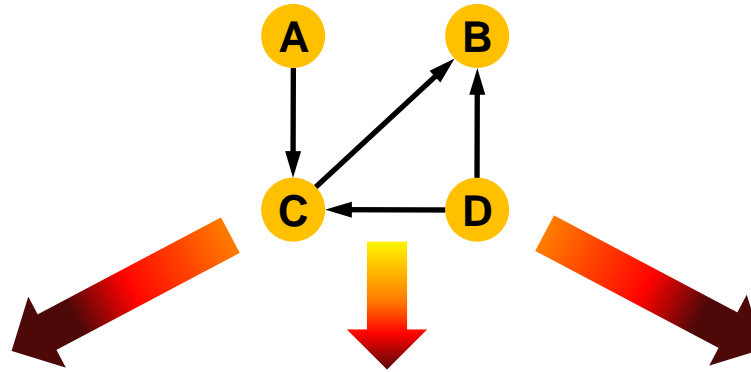
- We now have:
 - Shortest path from A to each node (both length and path)
 - **Minimum spanning tree**

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?

Computational Representation of Networks



List of edges:

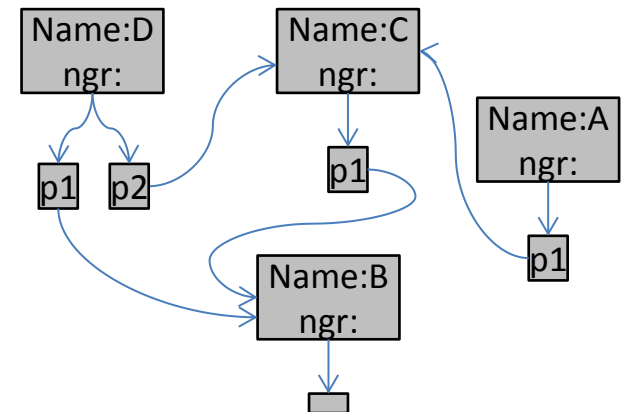
(ordered) pairs of nodes

[(A,C) , (C,B) ,
(D,B) , (D,C)]

Connectivity Matrix

	A	B	C	D
A	0	0	1	0
B	0	0	0	0
C	0	1	0	0
D	0	1	1	0

Object Oriented



- Which is the most useful representation?