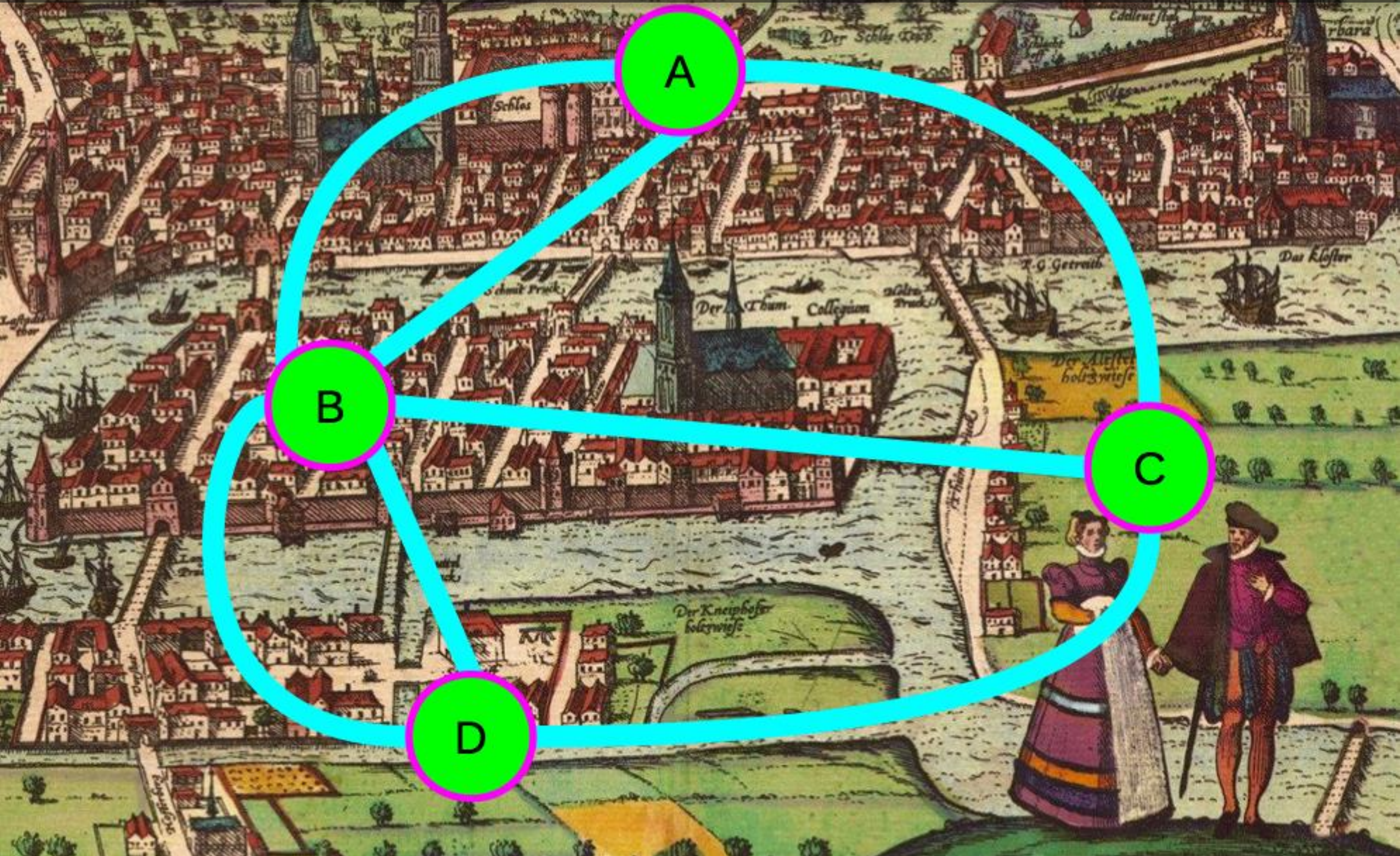
Graphs - I

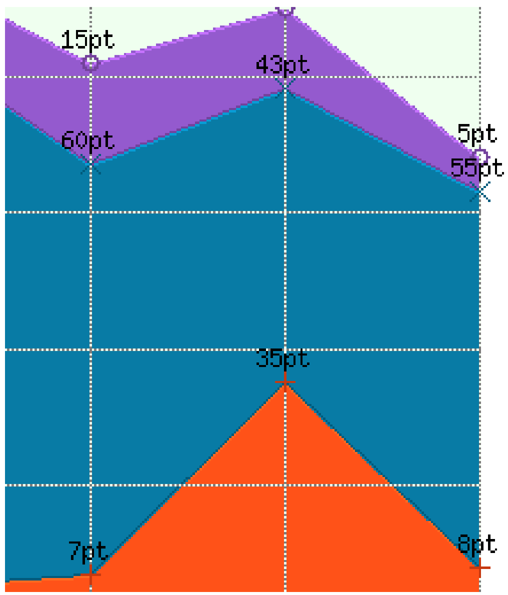
CS 2110, Spring 2016

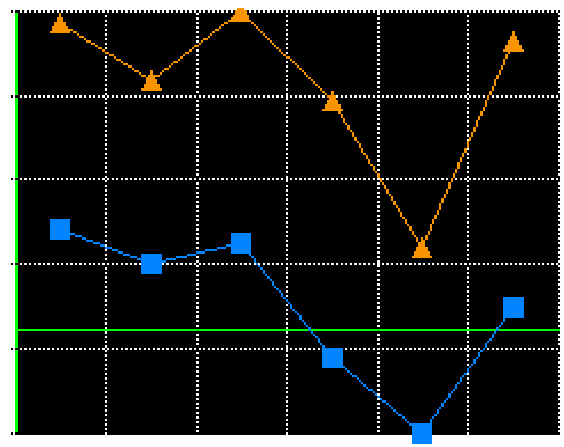


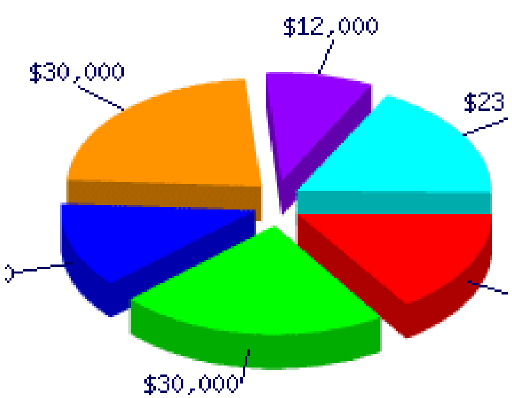
Announcements

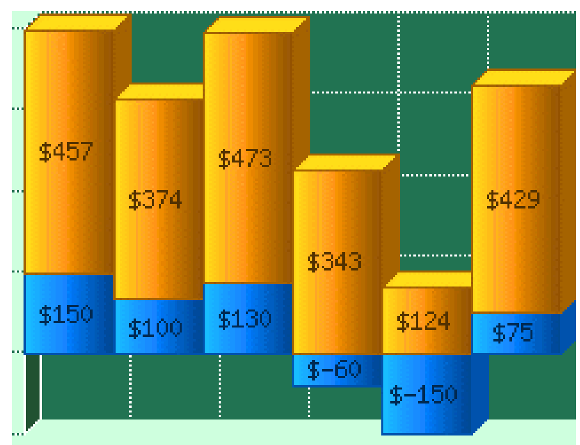
- Reading:
  - Chapter 28: Graphs
  - Chapter 29: Graph Implementations

These *aren’t* the graphs we’re interested in

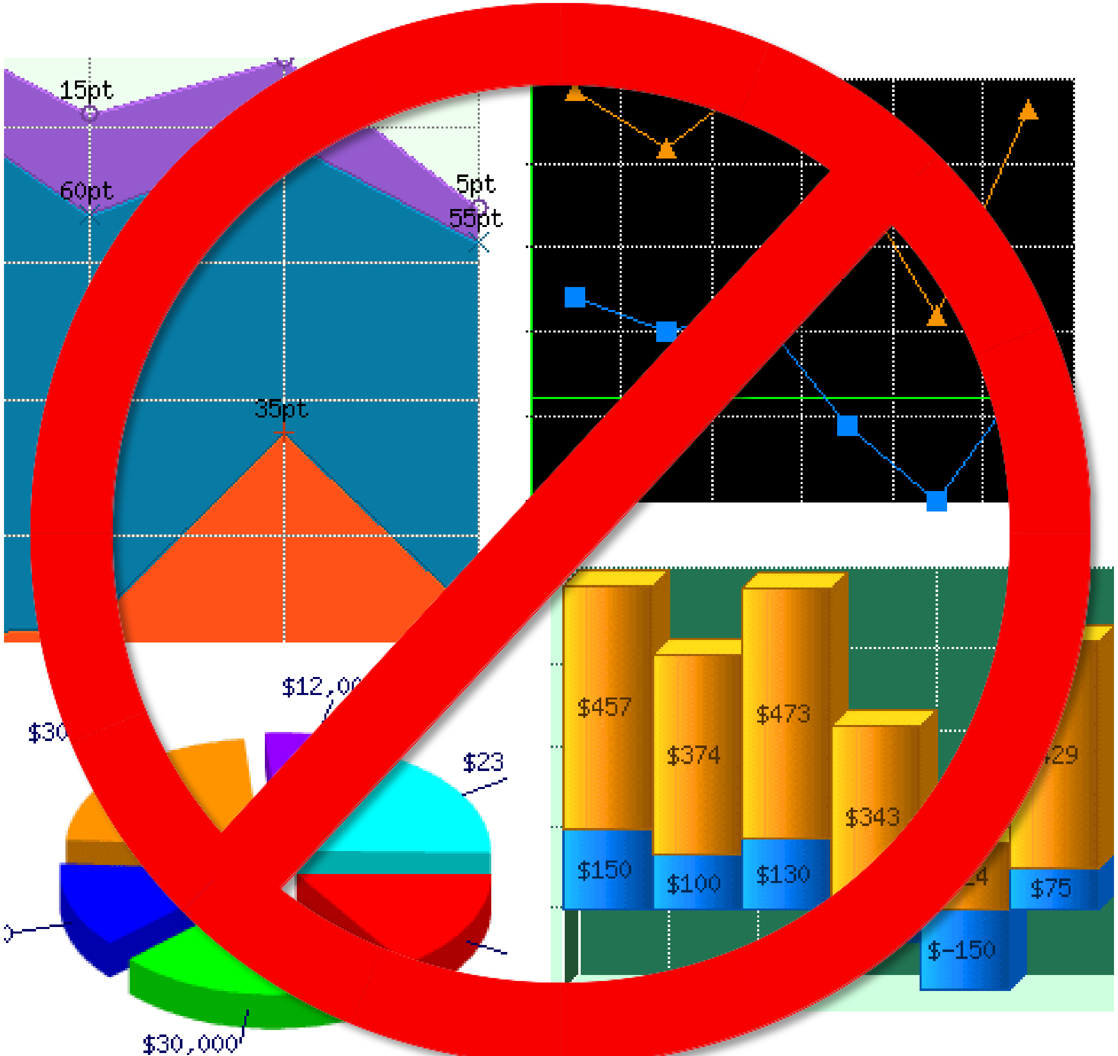




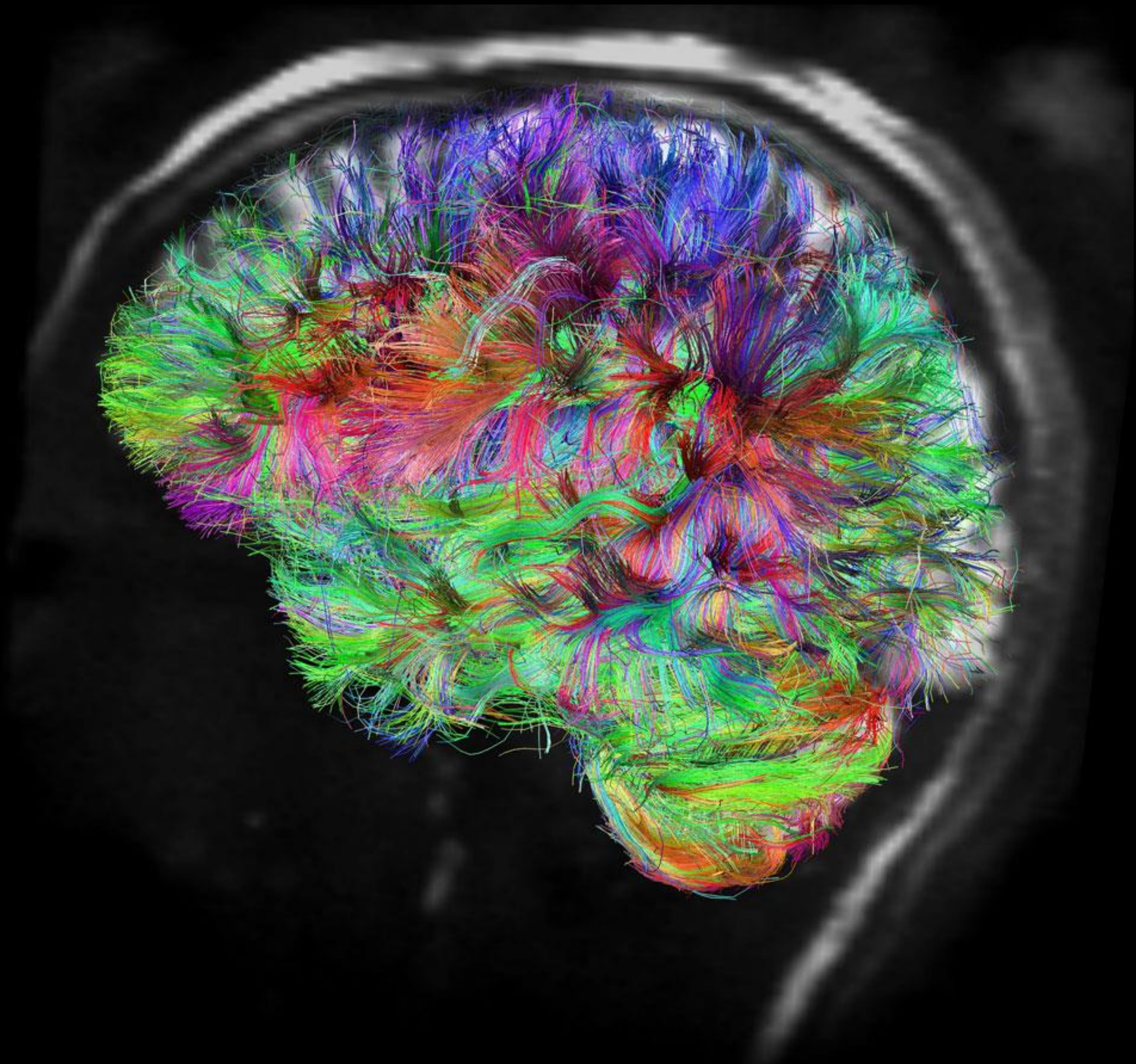




These *aren't* the graphs we're interested in

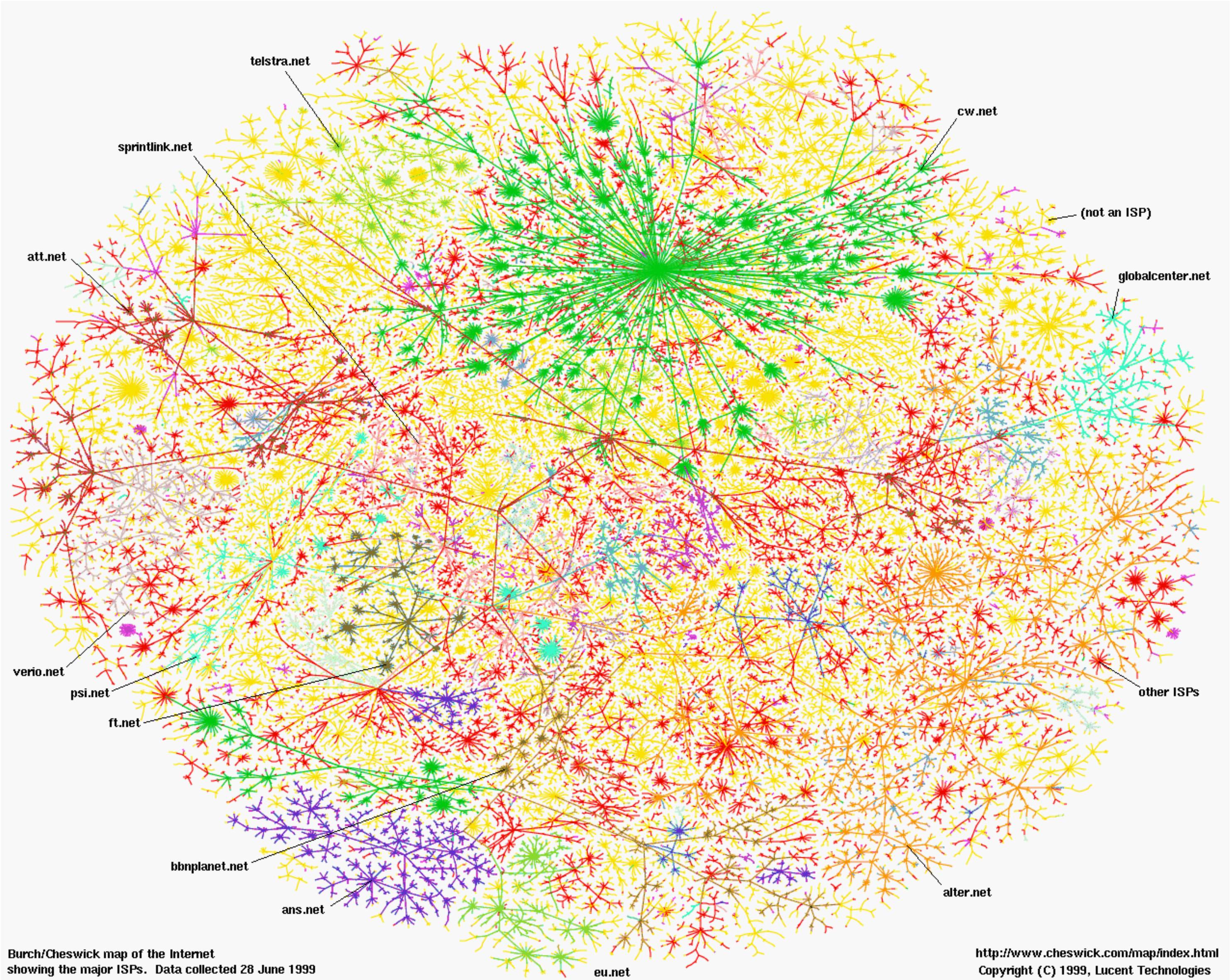


This is

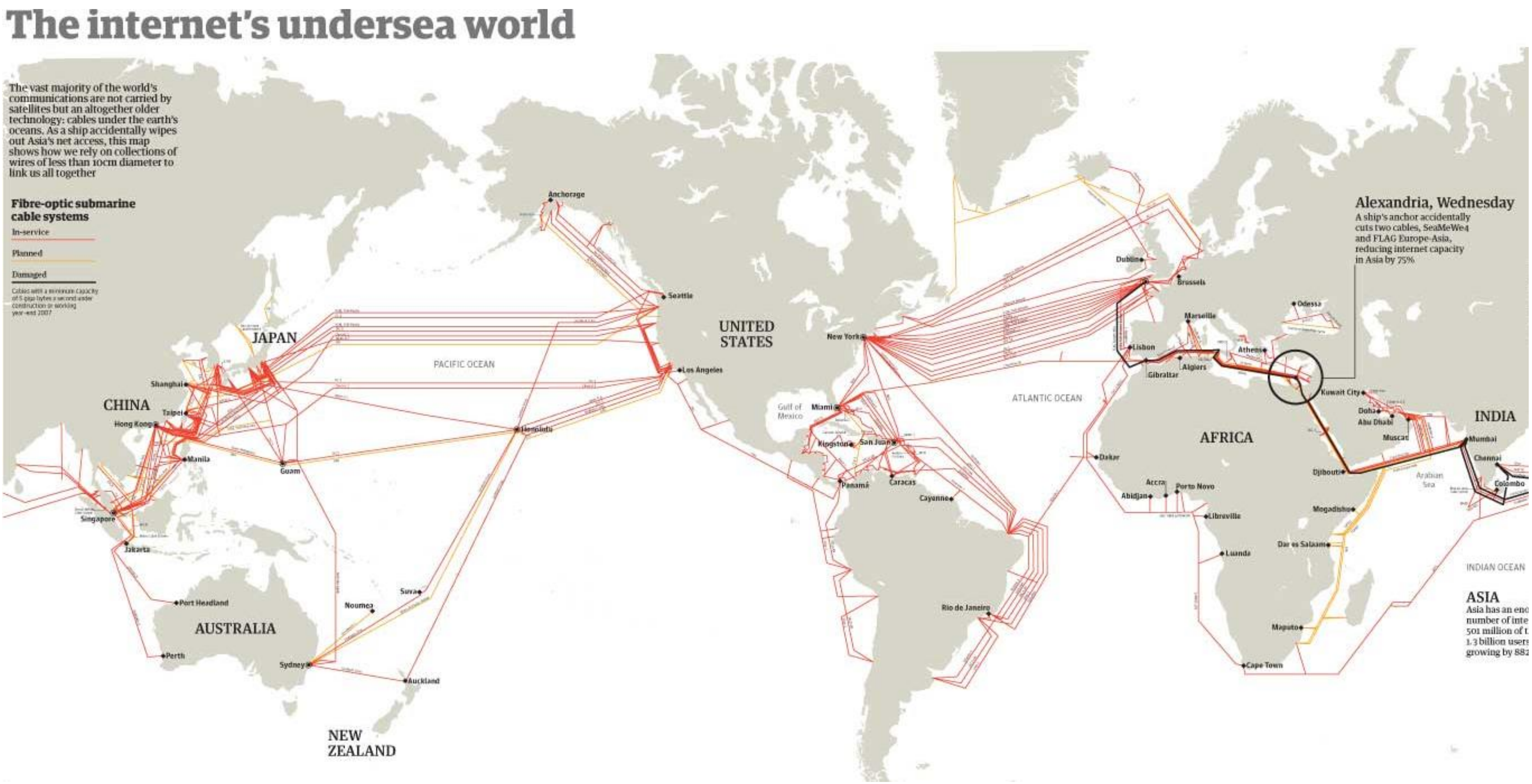


V.J. Wedeen and L.L. Wald, Martinos Center for Biomedical Imaging at MGH

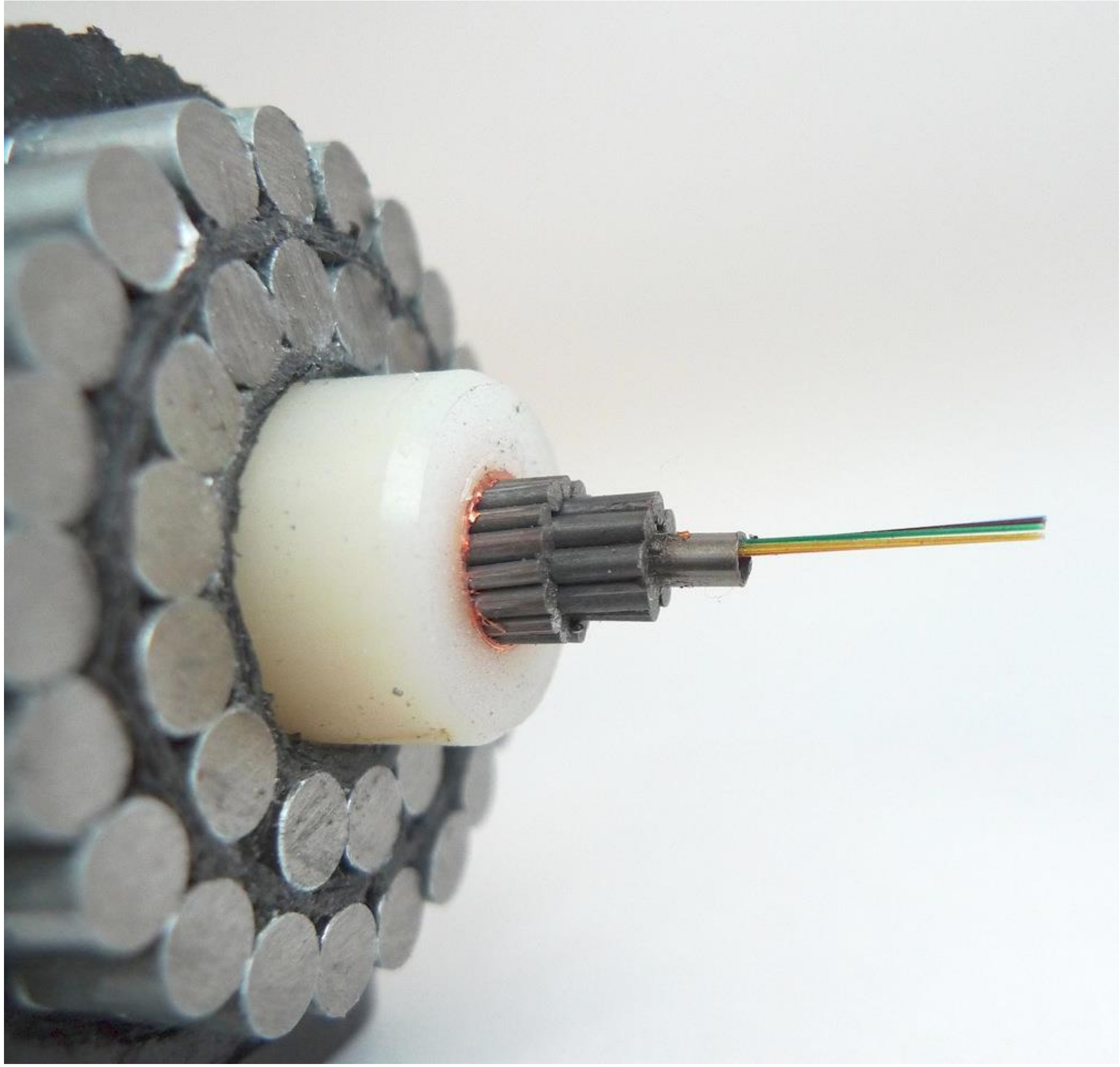
And so is this



And this


The internet's undersea world
The vast majority of the world's communications are not carried by satellites but an altogether older technology: cables under the earth's oceans. As a ship accidentally wipes out Asia's net access, this map shows how we rely on collections of wires of less than 10cm diameter to link us all together
Fibre-optic submarine cable systems
In-service
Planned
Damaged
Alexandria, Wednesday
A ship's anchor accidentally cuts two cables, SeaMeWe4 and FLAG Europe-Asia, reducing internet capacity in Asia by 75%
ASIA
CHINA
JAPAN
UNITED STATES
AFRICA
INDIA
AUSTRALIA
NEW ZEALAND
PACIFIC OCEAN
ATLANTIC OCEAN
INDIAN OCEAN
Gulf of Mexico
Arabian Sea
Anchorage
Seattle
Los Angeles
New York
Miami
Kingston
San Juan
Panamá
Caracas
Cayenne
Rio de Janeiro
Honolulu
Guam
Shanghai
Taipei
Hong Kong
Manila
Singapore
Jakarta
Port Hedland
Perth
Sydney
Noumea
Suva
Auckland
Dublin
Brussels
Marseille
Lisbon
Gibraltar
Algiers
Athens
Odessa
Kuwait City
Doha
Abu Dhabi
Muscat
Mumbai
Chennai
Colombo
Djibouti
Mogadishu
Dar es Salaam
Maputo
Cape Town
Luanda
Libreville
Accra
Porto Novo
Abidjan
Dakar

This carries Internet traffic across the oceans

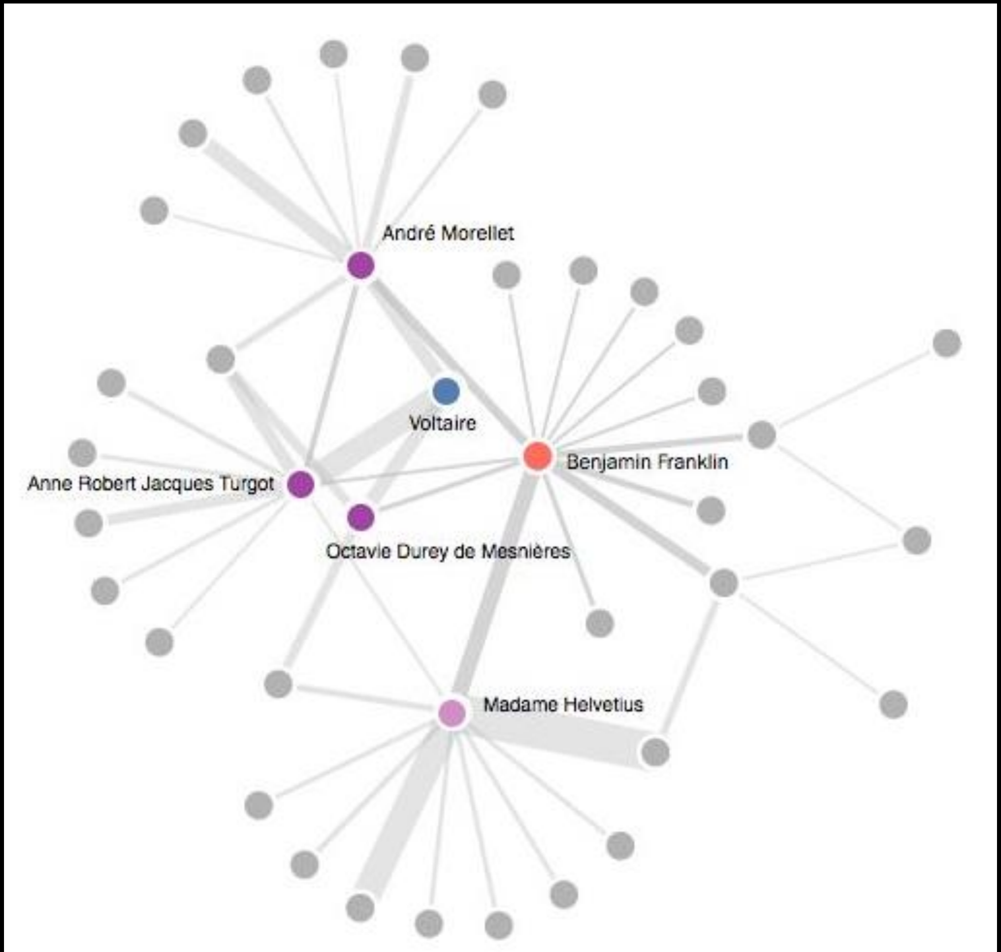


A social graph

An older social graph

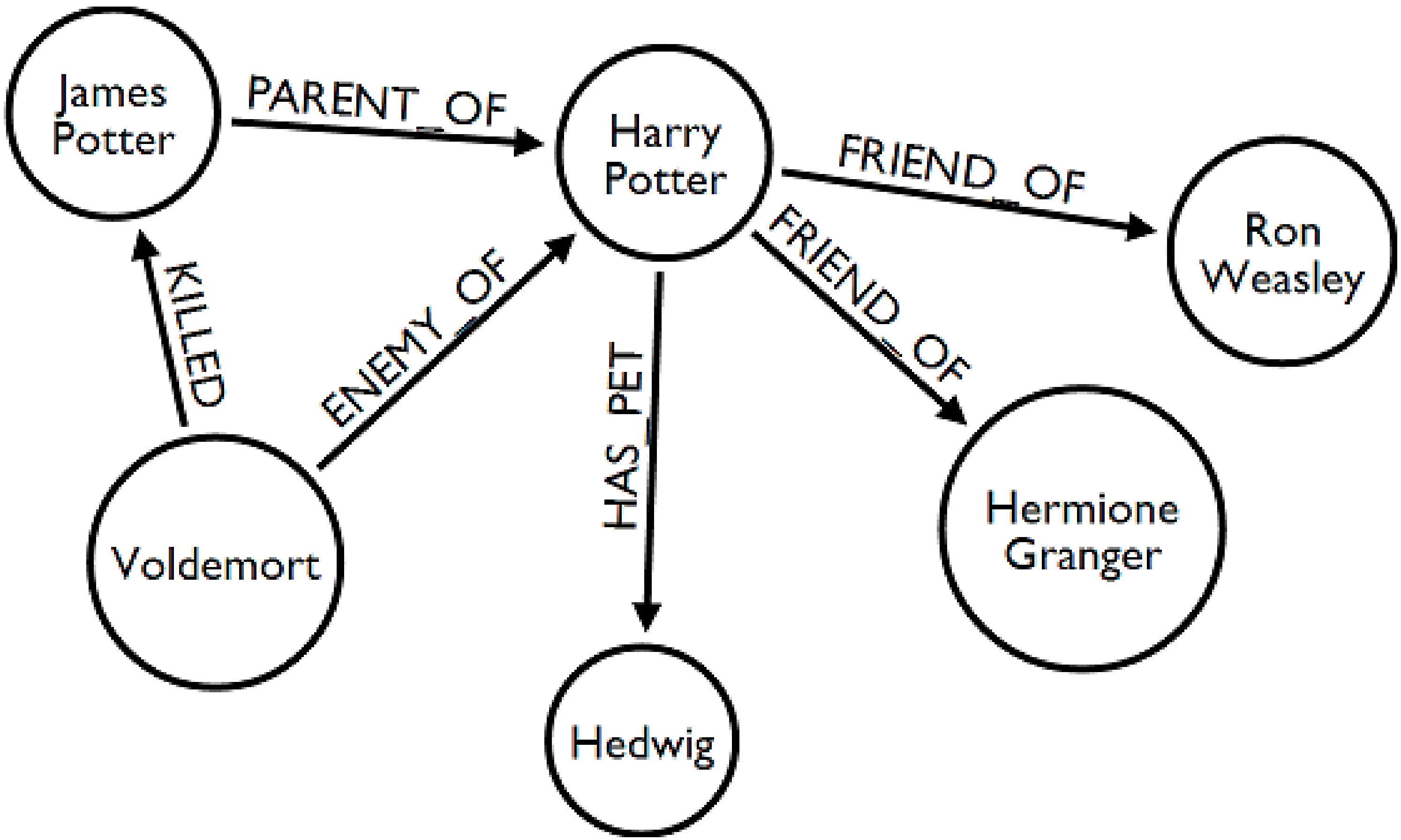
Locke's (blue) and Voltaire's (yellow) correspondence.
Only letters for which complete location information is available are shown.
Data courtesy the Electronic Enlightenment Project, University of Oxford.

An older social graph

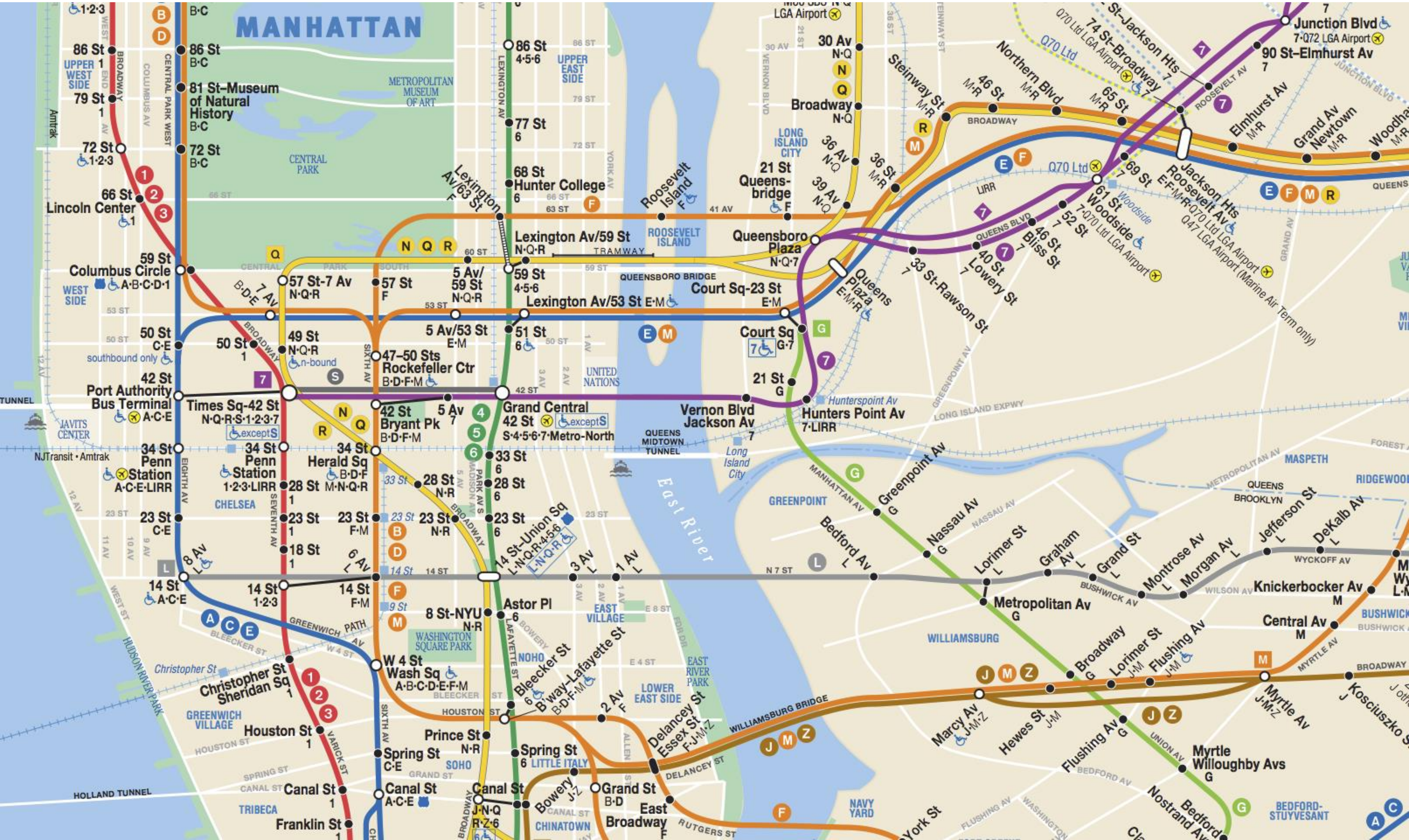


Voltaire and Benjamin Franklin

A fictional social graph

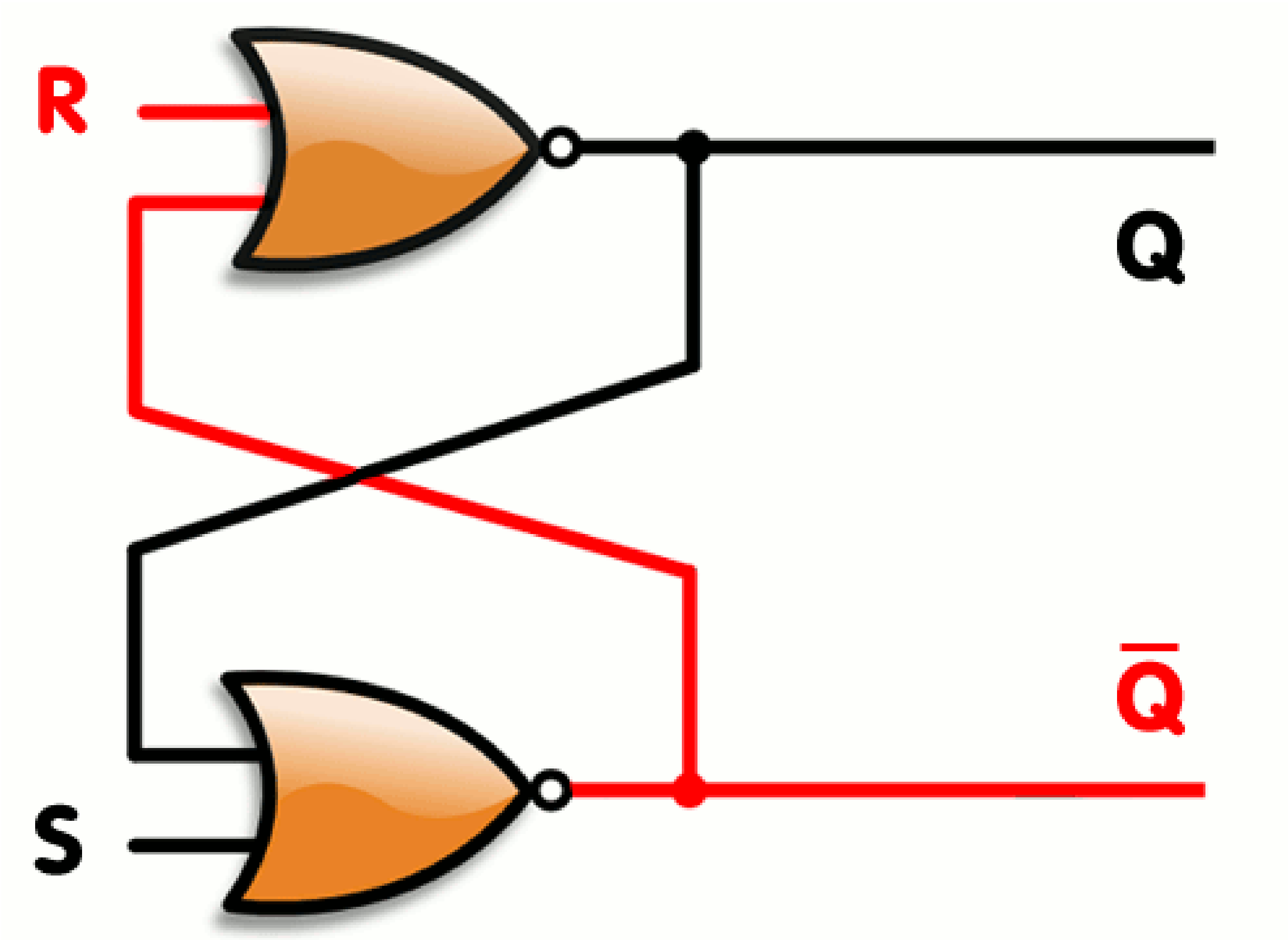


A transport graph

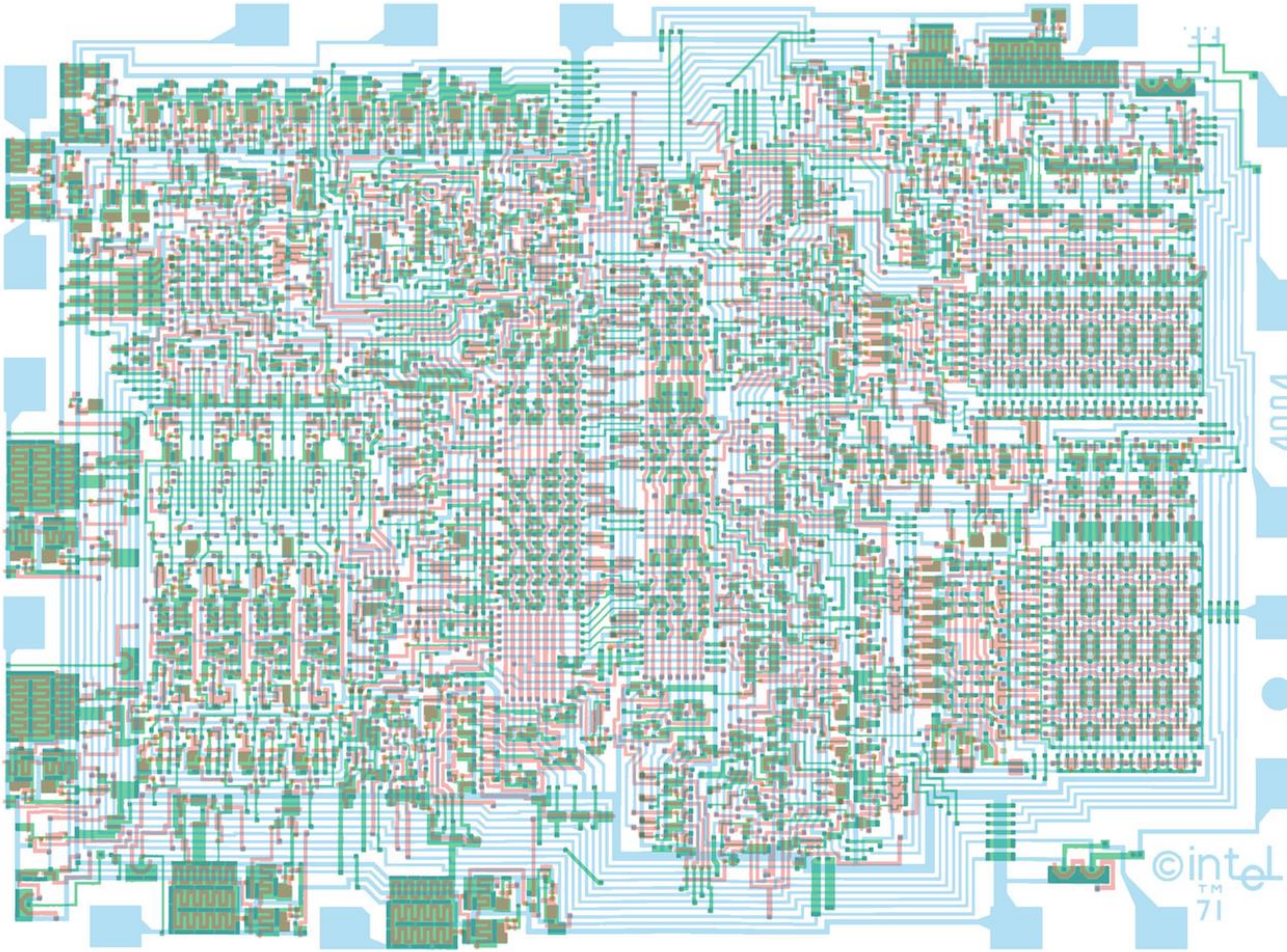


Another transport graph

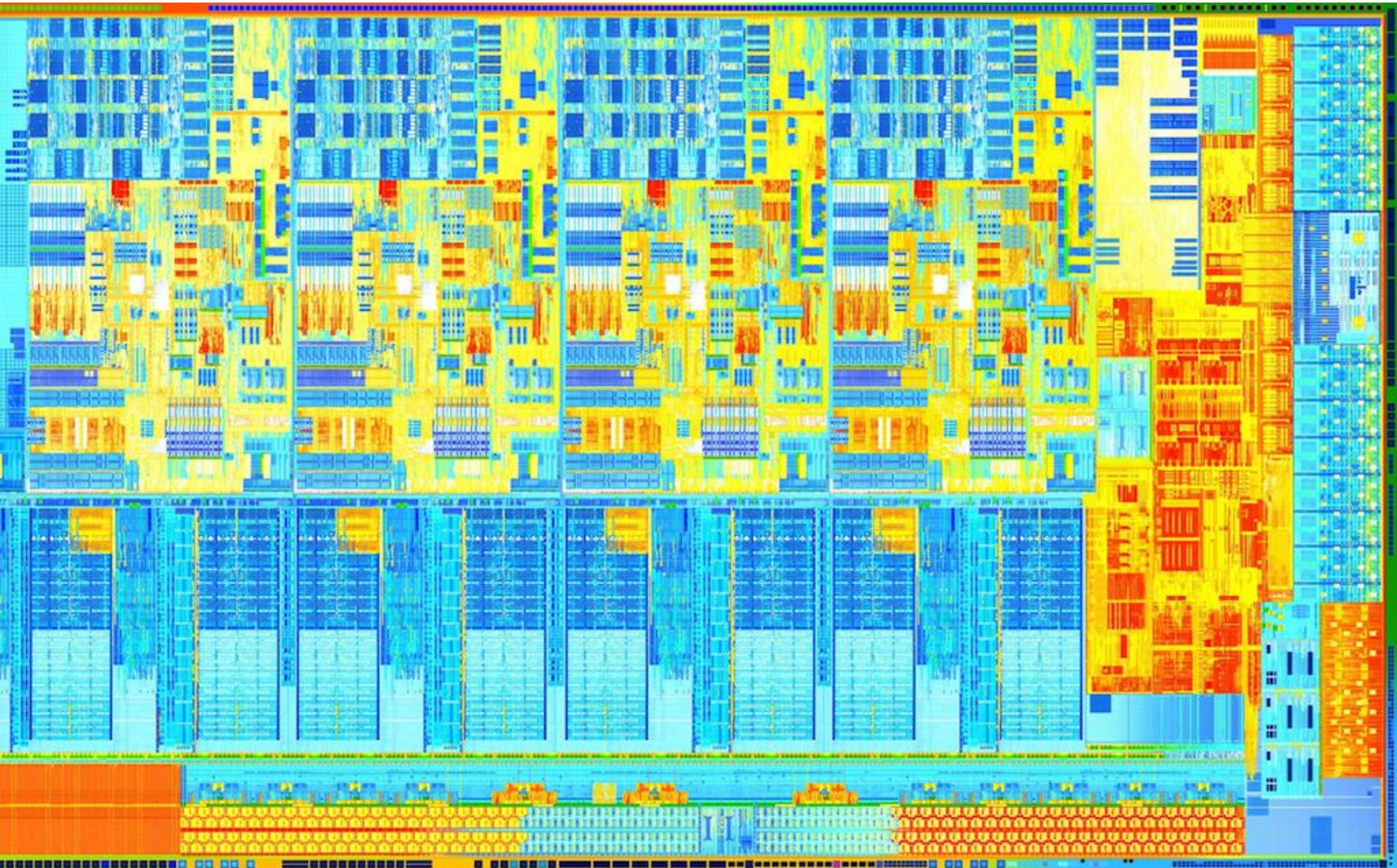
A circuit graph (flip-flop)



A circuit graph (Intel 4004)

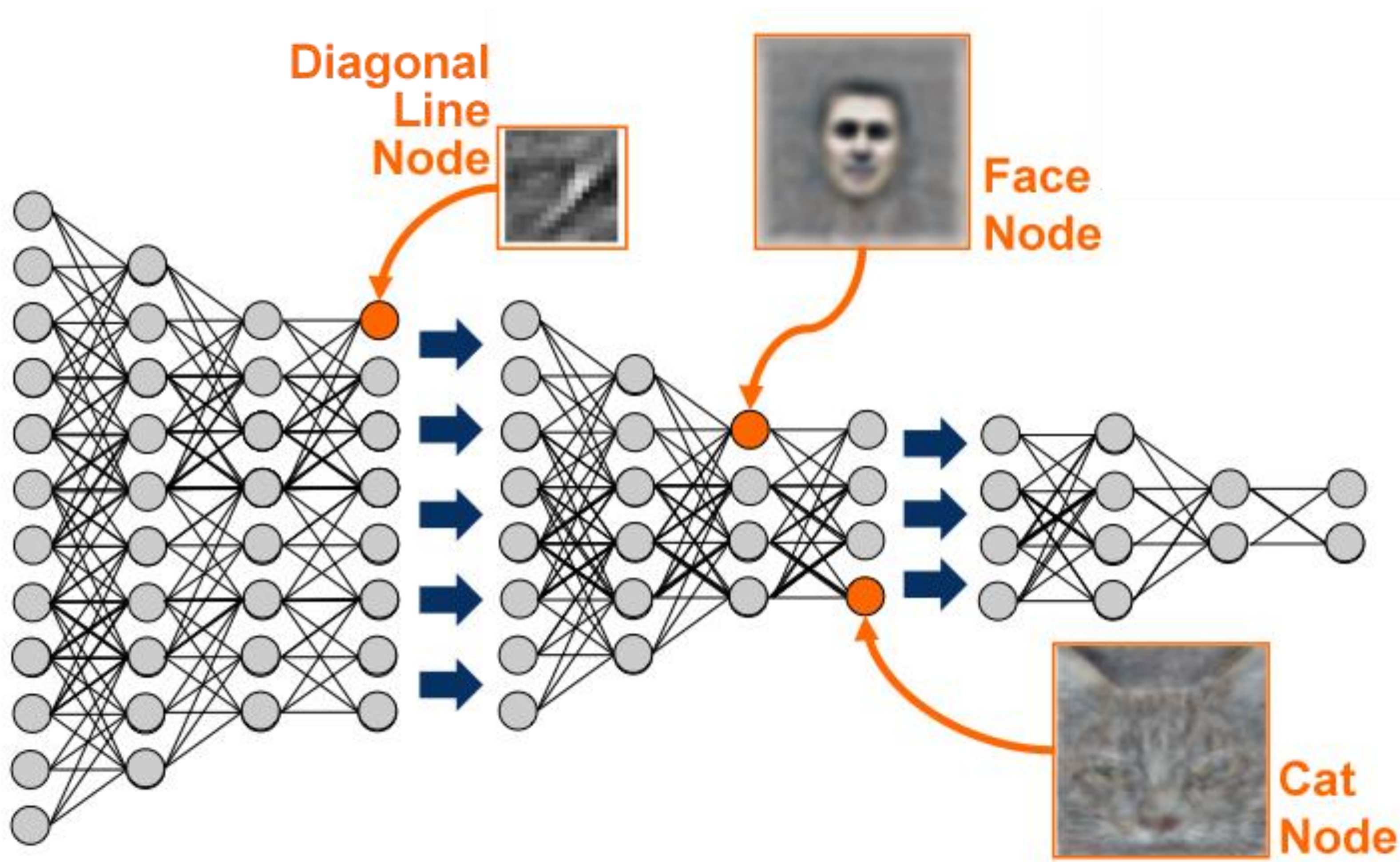


A circuit graph (Intel Haswell)

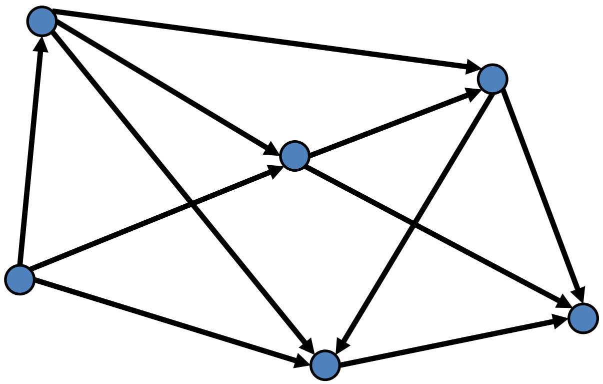


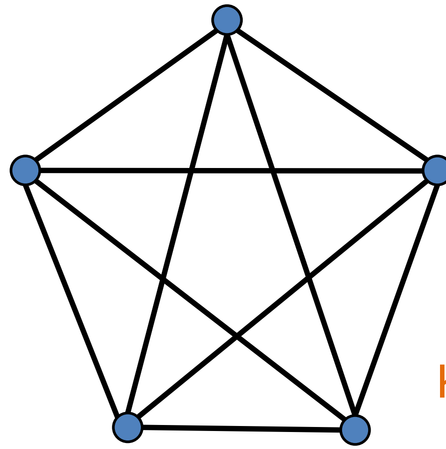
This is not a graph, this is a cat

This is a graph(ical model) that has learned to recognize cats

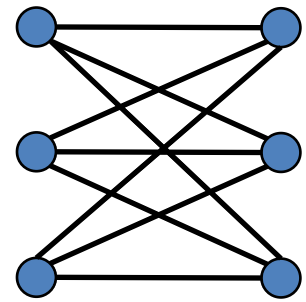


Some abstract graphs

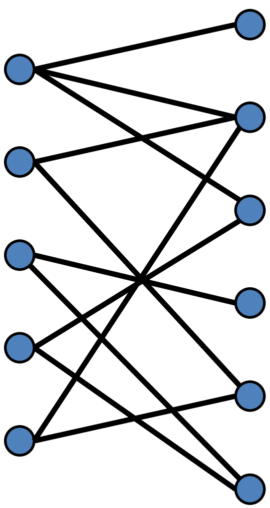


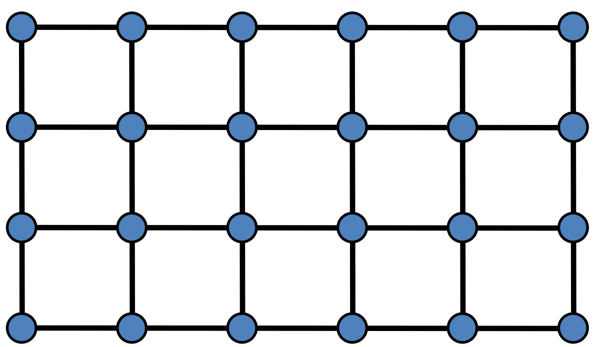


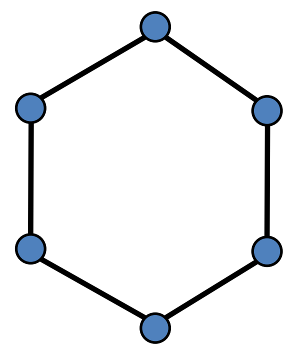
K_5



$K_{3,3}$





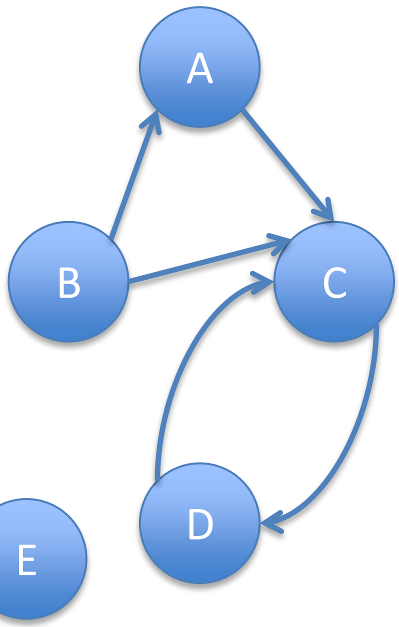


Directed Graphs

- A directed graph (digraph) is a pair (V, E) where
  - V is a (finite) set
  - E is a set of ***ordered*** pairs (u, v) where $u,v \in V$
    - Often require $u \neq v$ (i.e. no self-loops)

- An element of V is called a vertex or node
- An element of E is called an edge or arc

- $|V|$ = size of V, often denoted by n
- $|E|$ = size of E, often denoted by m



$\boldsymbol{V} = \{A, B, C, D, E\}$

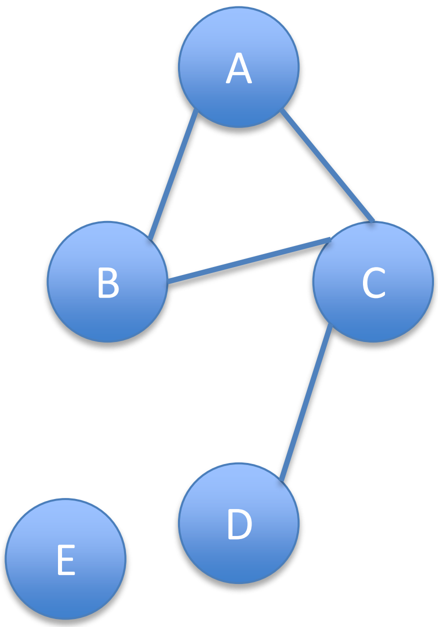
$\boldsymbol{E} = \{(A,C), (B,A), (B,C), (C,D), (D,C)\}$

$|\boldsymbol{V}| = 5$

$|\boldsymbol{E}| = 5$

Undirected Graphs

- An undirected graph is just like a directed graph!
  - … except that E is now a set of ***unordered*** pairs $\{u, v\}$ where $u,v \in V$
- Every undirected graph can be easily converted to an equivalent directed graph via a simple transformation:
  - Replace every undirected edge with two directed edges in opposite directions
- … but not vice versa



$\boldsymbol{V} = \{A, B, C, D, E\}$

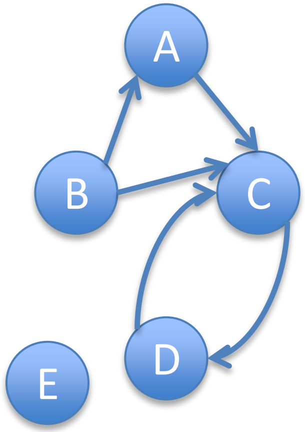
$\boldsymbol{E} = \{\{A,C\}, \{B,A\}, \{B,C\}, \{C,D\}\}$

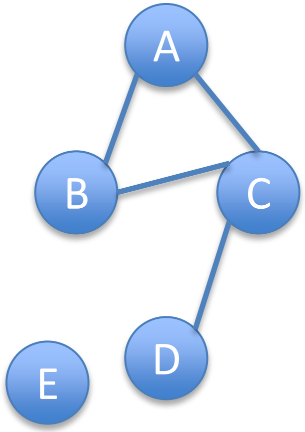
$|\boldsymbol{V}| = 5$

$|\boldsymbol{E}| = 4$

Graph Terminology

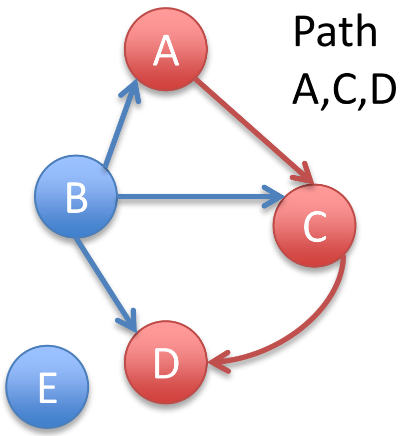
- Vertices u and v are called
  - the source and sink of the directed edge (u, v), respectively
  - the endpoints of (u, v) or $\{u, v\}$
- Two vertices are adjacent if they are connected by an edge
- The outdegree of a vertex u in a directed graph is the number of edges for which u is the source
- The indegree of a vertex v in a directed graph is the number of edges for which v is the sink
- The degree of a vertex u in an undirected graph is the number of edges of which u is an endpoint



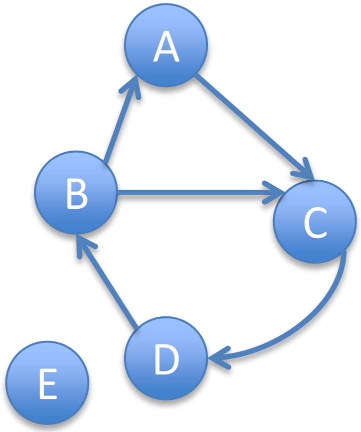


More Graph Terminology

- A path is a sequence $v_0, v_1, v_2, \ldots, v_p$ of vertices such that for $0 \leq i < p$,
  - $(v_i, v_{i+1}) \in E$ if the graph is directed
  - $\{v_i, v_{i+1}\} \in E$ if the graph is undirected
- The length of a path is its number of edges
  - In this example, the length is 2
- A path is simple if it doesn't repeat any vertices
- A cycle is a path $v_0, v_1, v_2, \ldots, v_p$ such that $v_0 = v_p$
- A cycle is simple if it does not repeat any vertices except the first and last
- A graph is acyclic if it has no cycles
- A ***d***irected ***a***cyclic ***g***raph is called a DAG

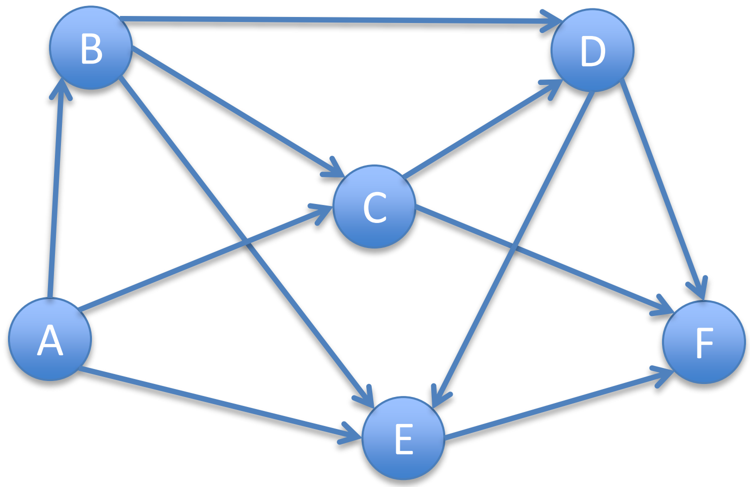


DAG



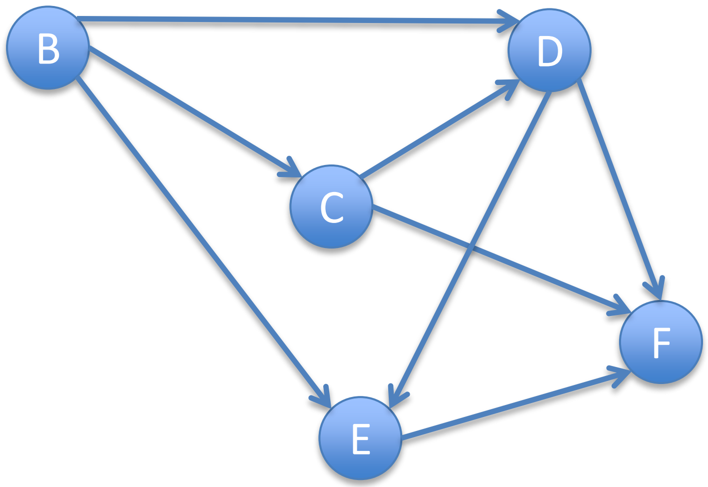
Not a DAG

Is this a DAG?



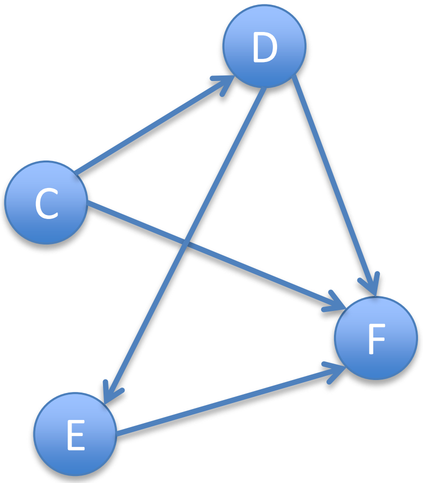
- Intuition:
  - If it's a DAG, there must be a vertex with indegree zero
- This idea leads to an *algorithm*
  - A digraph is a DAG if and only if we can iteratively delete indegree-0 vertices until the graph disappears

Is this a DAG?



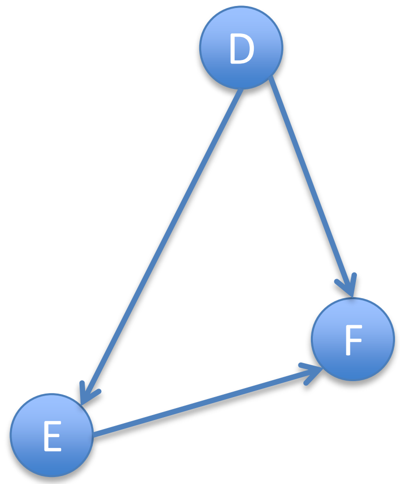
- Intuition:
  - If it's a DAG, there must be a vertex with indegree zero
- This idea leads to an *algorithm*
  - A digraph is a DAG if and only if we can iteratively delete indegree-0 vertices until the graph disappears

Is this a DAG?



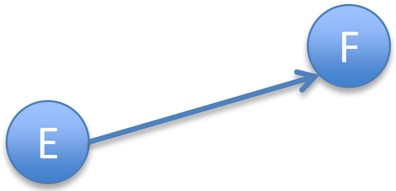
- Intuition:
  - If it's a DAG, there must be a vertex with indegree zero
- This idea leads to an *algorithm*
  - A digraph is a DAG if and only if we can iteratively delete indegree-0 vertices until the graph disappears

Is this a DAG?



- Intuition:
  - If it's a DAG, there must be a vertex with indegree zero
- This idea leads to an *algorithm*
  - A digraph is a DAG if and only if we can iteratively delete indegree-0 vertices until the graph disappears

Is this a DAG?



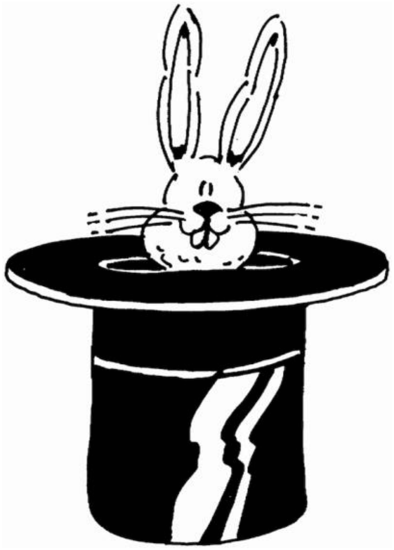
- Intuition:
  - If it's a DAG, there must be a vertex with indegree zero
- This idea leads to an *algorithm*
  - A digraph is a DAG if and only if we can iteratively delete indegree-0 vertices until the graph disappears

Is this a DAG?

- Intuition:
  - If it's a DAG, there must be a vertex with indegree zero
- This idea leads to an *algorithm*
  - A digraph is a DAG if and only if we can iteratively delete indegree-0 vertices until the graph disappears

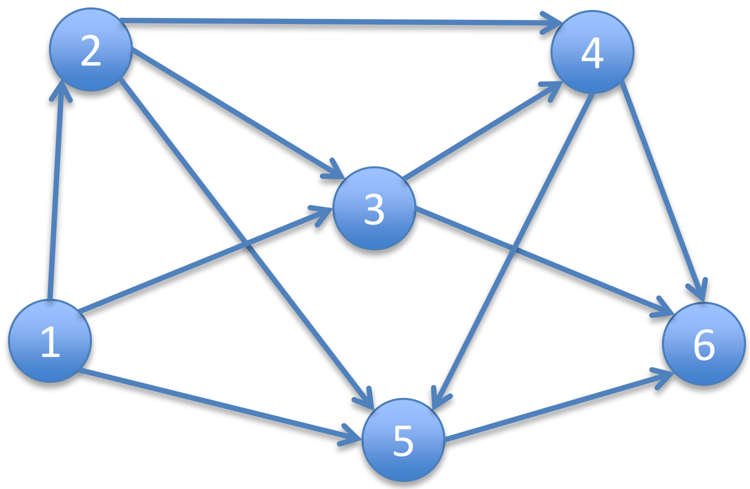
Is this a DAG?

YES!



- Intuition:
  - If it's a DAG, there must be a vertex with indegree zero
- This idea leads to an *algorithm*
  - A digraph is a DAG if and only if we can iteratively delete indegree-0 vertices until the graph disappears

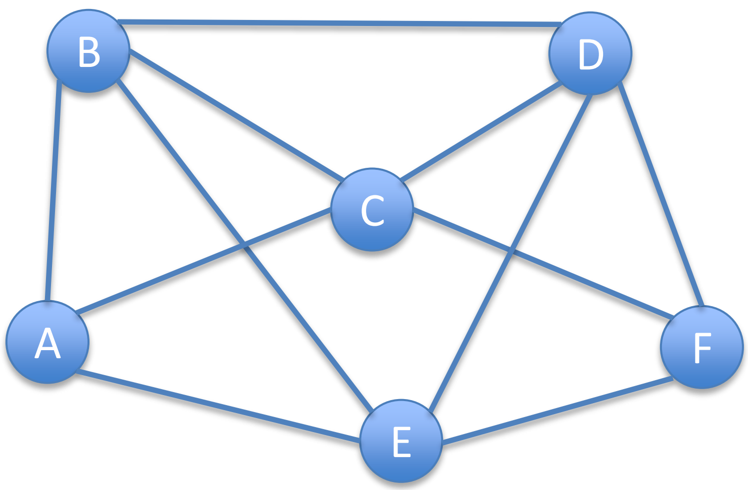
Topological Sort



- We just computed a topological sort of the DAG
  - This is a numbering of the vertices such that all edges go from lower- to higher-numbered vertices
  - Useful in job scheduling with precedence constraints

Graph Coloring

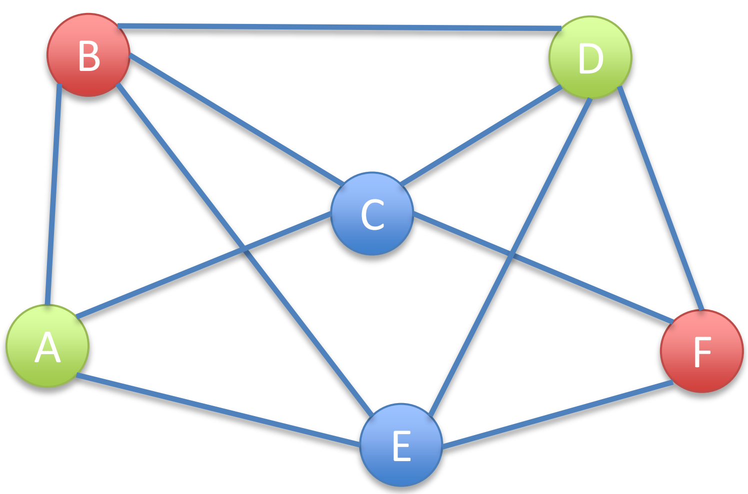
- A coloring of an undirected graph is an assignment of a color to each node such that no two adjacent vertices get the same color



- How many colors are needed to color this graph?

Graph Coloring

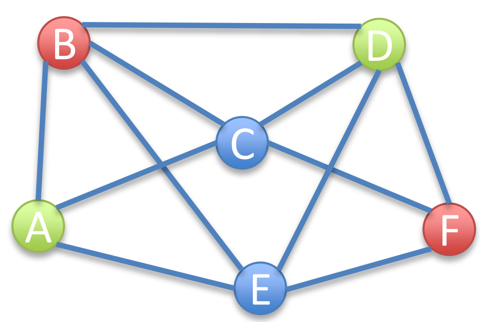
- A coloring of an undirected graph is an assignment of a color to each node such that no two adjacent vertices get the same color



- How many colors are needed to color this graph?

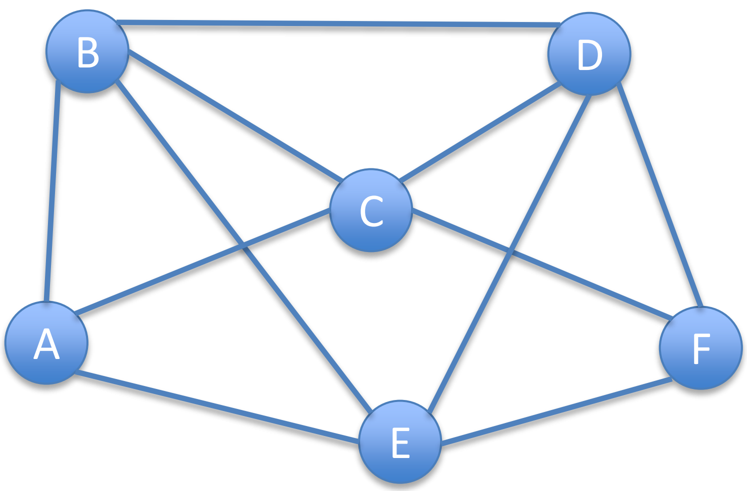
An Application of Coloring

- Vertices are **tasks**
- Edge (u, v) is present if tasks u and v each require access to the **same shared resource**, and thus cannot execute simultaneously
- Colors are **time slots** to schedule the tasks
- Minimum number of colors needed to color the graph = minimum number of time slots required



Planarity

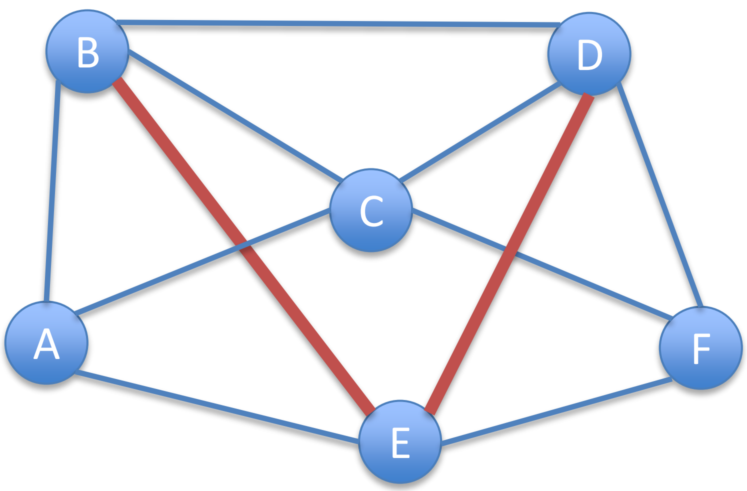
- A graph is planar if it can be drawn in the plane without any edges crossing



- Is this graph planar?

Planarity

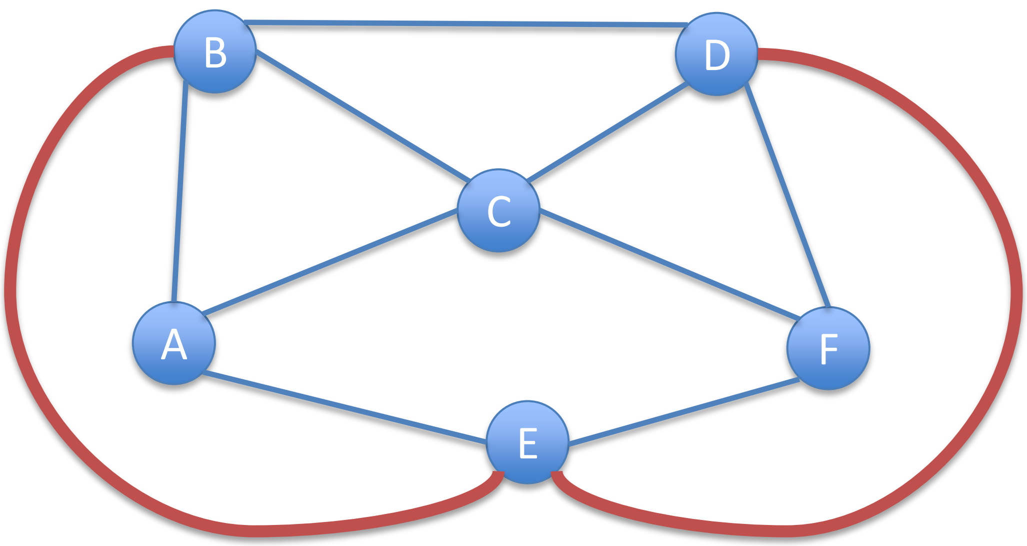
- A graph is planar if it can be drawn in the plane without any edges crossing



- Is this graph planar?
  - Yes!

Planarity

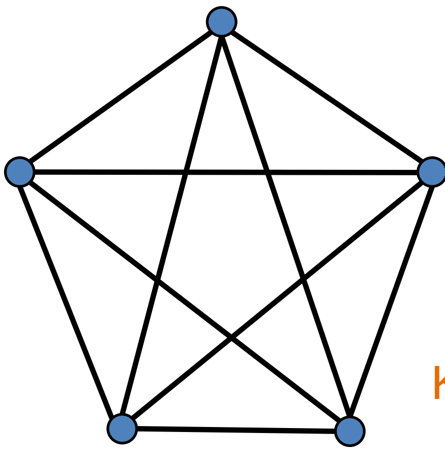
- A graph is planar if it **can** be drawn in the plane without any edges crossing



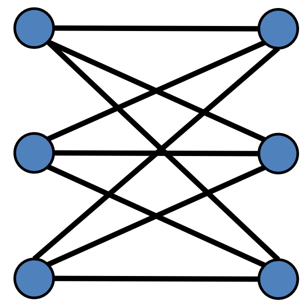
- Is this graph planar?
  - Yes!

Detecting Planarity

Kuratowski's Theorem:



K_5



$K_{3,3}$

- A graph is planar if and only if it does not contain a copy of K_5 or $K_{3,3}$ (possibly with other nodes along the edges shown)

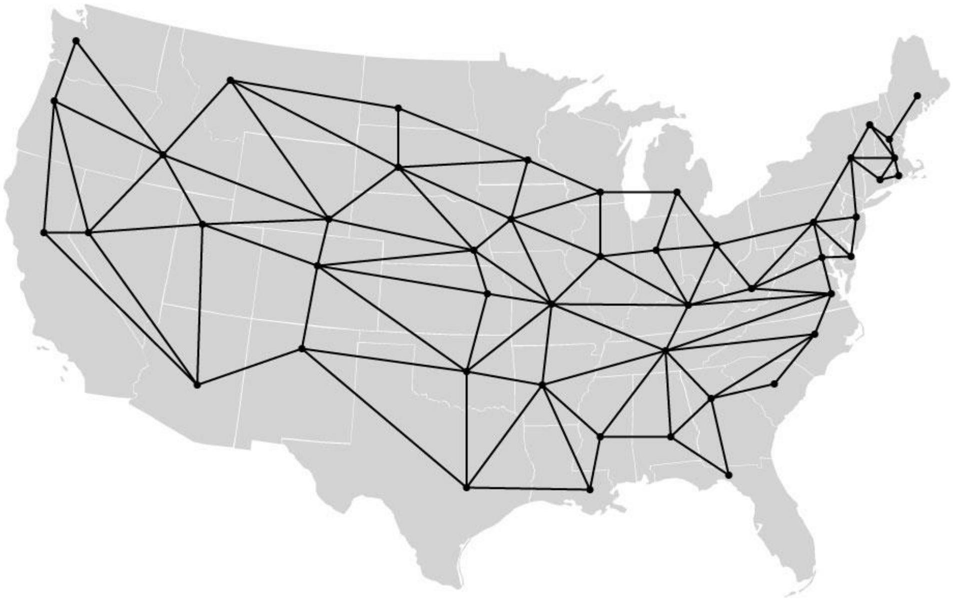
Four-Color Theorem:

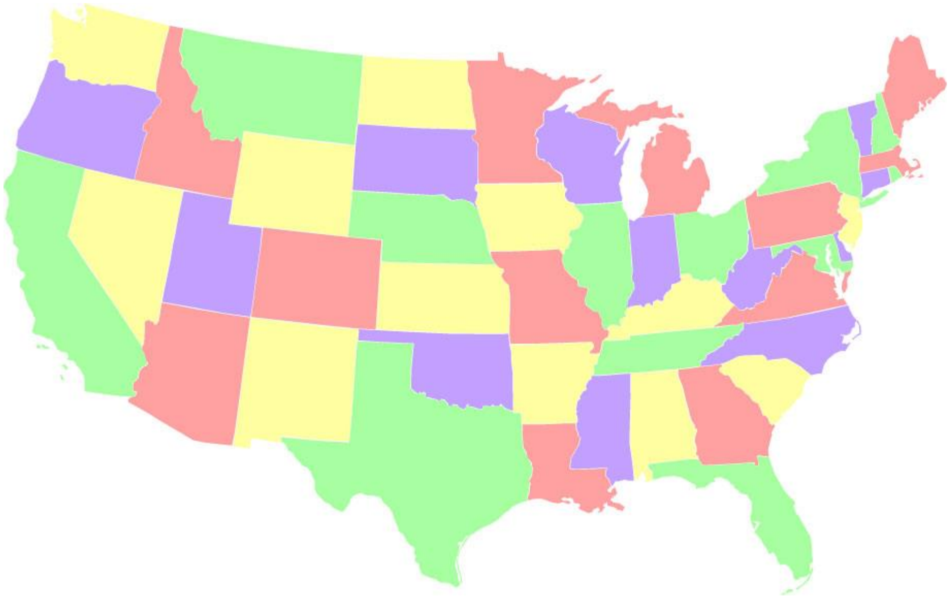
Every planar graph is 4-colorable

[Appel & Haken, 1976]

(Every map defines a planar graph – countries are vertices, and two adjacent countries define an edge)

Another 4-colored planar graph



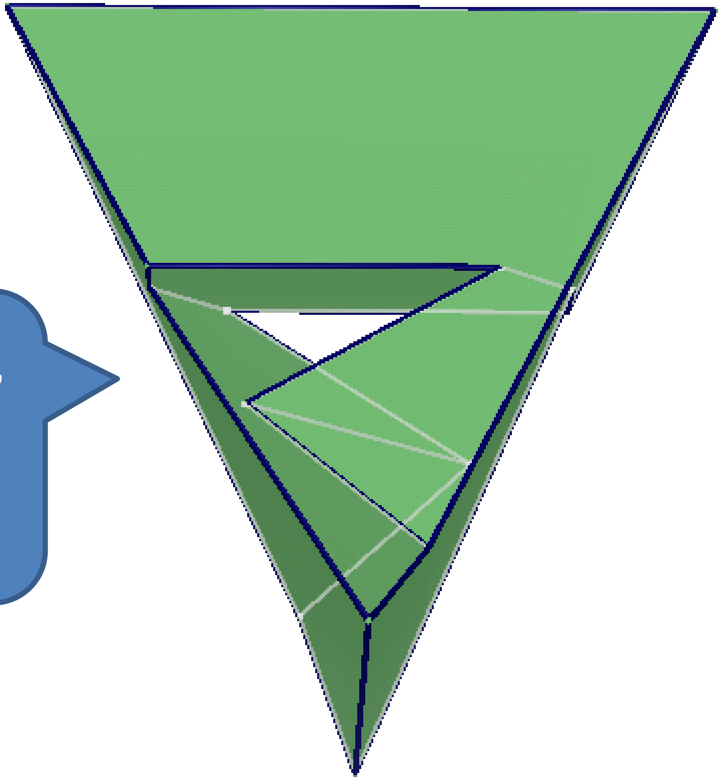


http://www.cs.cmu.edu/~bryant/boolean/maps.html

Szilassi polyhedron

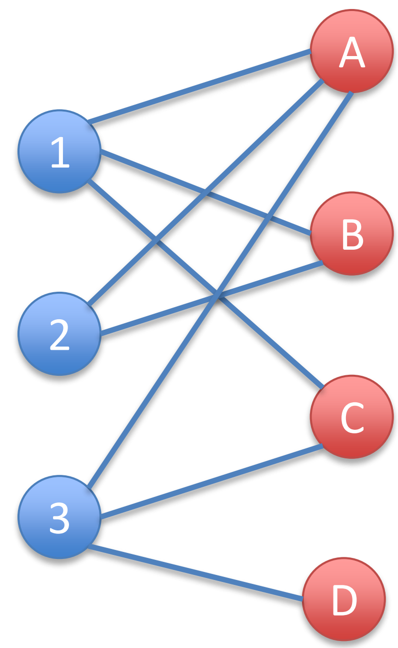
Torus (donut) maps are always 7-colorable

Has 7 hexagonal faces, all of which border every other face



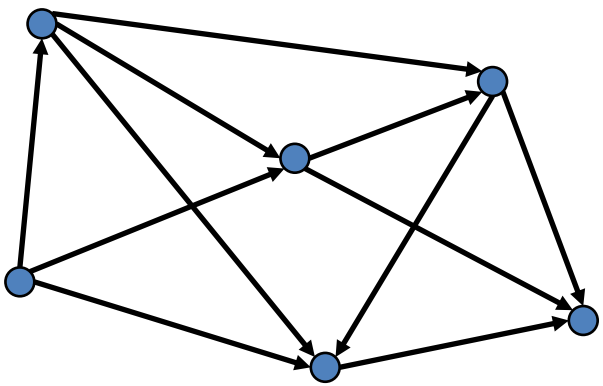
Bipartite Graphs

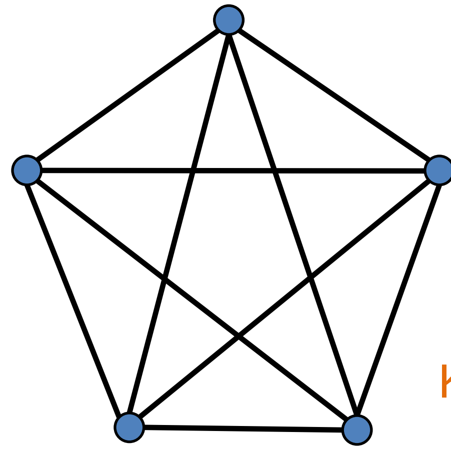
- A directed or undirected graph is bipartite if the vertices can be partitioned into two sets such that no edge connects two vertices in the same set



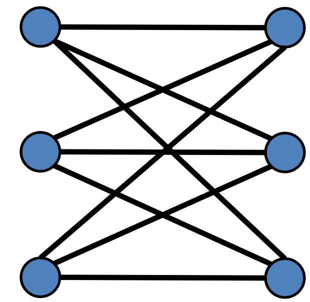
- The following are equivalent
  - G is bipartite
  - G is 2-colorable
  - G has no cycles of odd length

Some abstract graphs

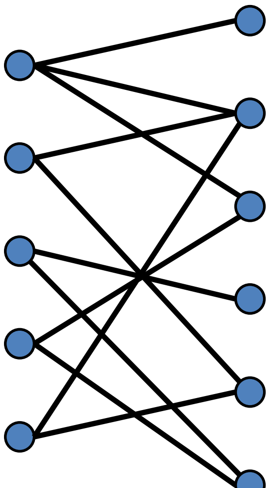


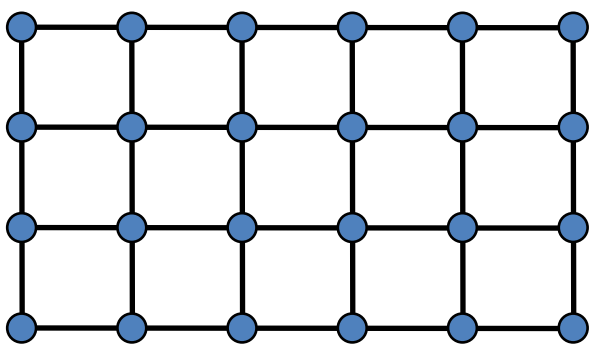


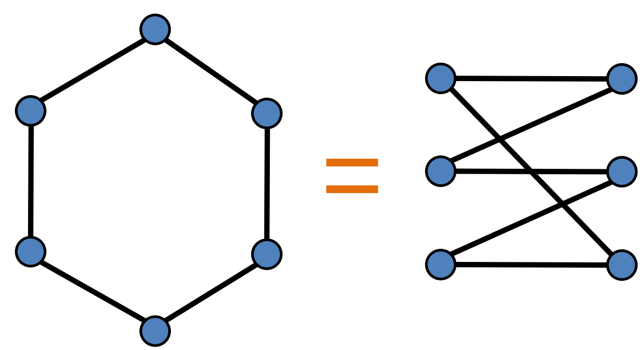
K_5



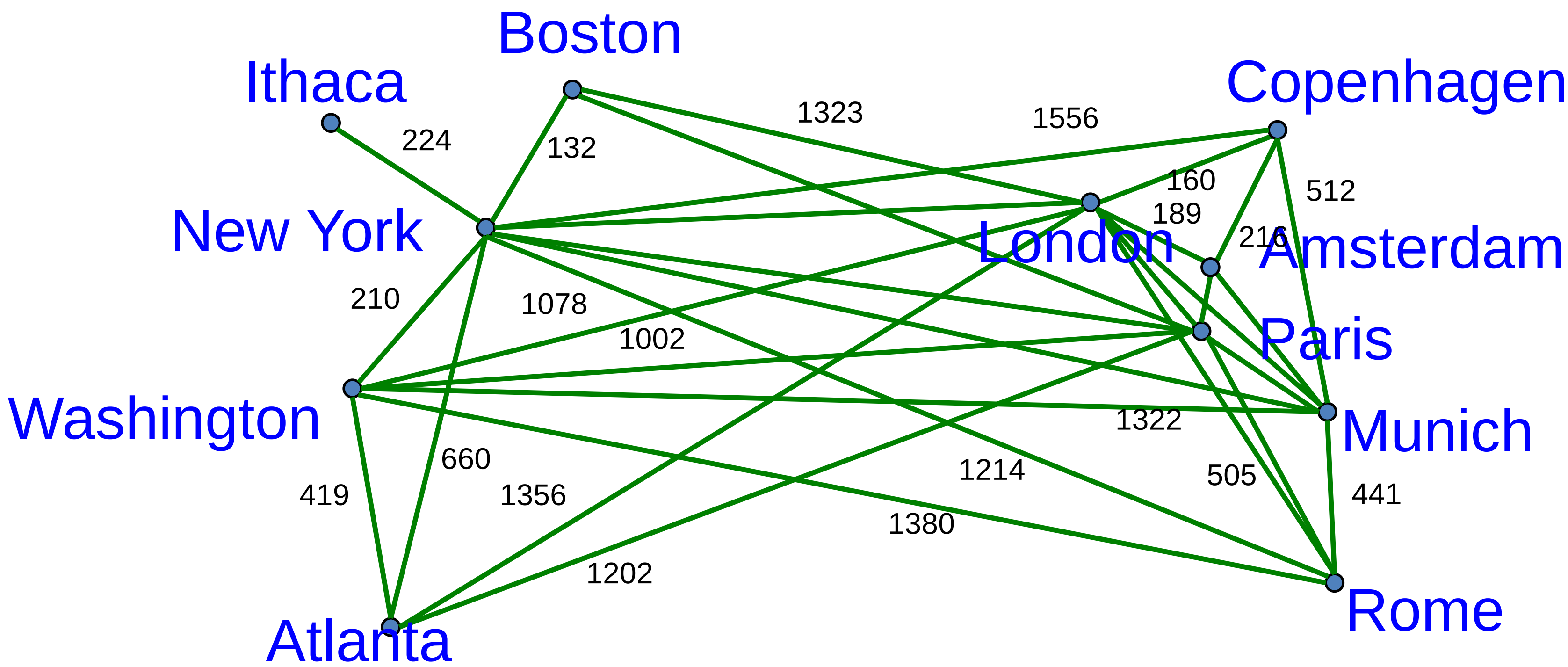
$K_{3,3}$





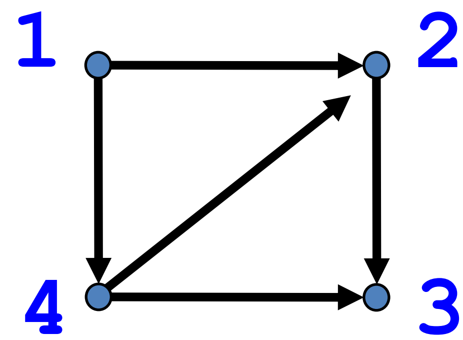


Traveling Salesperson

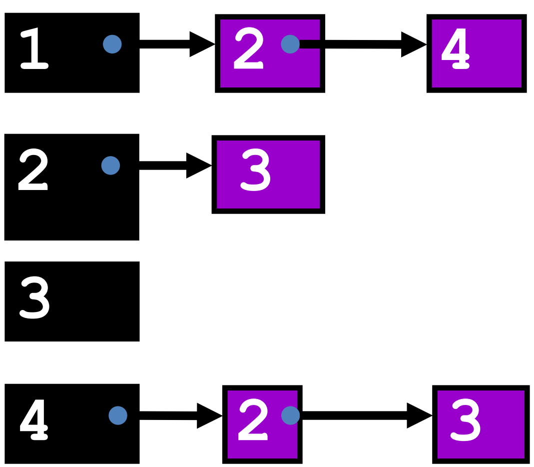


Find a path of minimum distance that visits every city

Representations of Graphs



Adjacency List



Adjacency Matrix

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

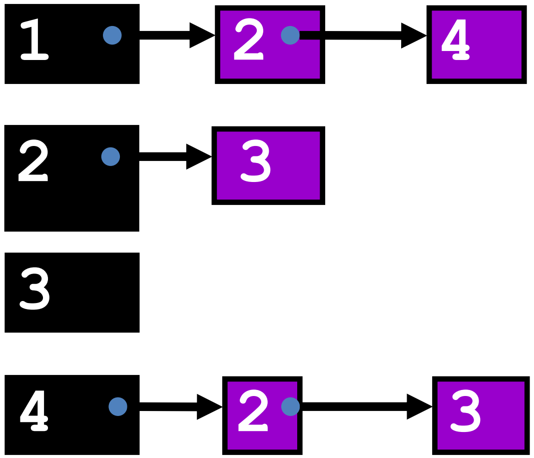
Adjacency Matrix or Adjacency List?

- n = number of vertices
- m = number of edges
- $d(u)$ = degree of u = no. of edges leaving u

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

- Adjacency Matrix
  - Uses space $O(n^2)$
  - Enumerate all edges in time $O(n^2)$
  - Answer "Is there an edge from u to v?" in $O(1)$ time
  - Better for dense graphs (lots of edges)

Adjacency Matrix or Adjacency List?



- n = number of vertices
- m = number of edges
- $d(u)$ = degree of u = no. edges leaving u

- Adjacency List
  - Uses space $O(m + n)$
  - Enumerate all edges in time $O(m + n)$
  - Answer "Is there an edge from u to v?" in $O(d(u))$ time
  - Better for sparse graphs (fewer edges)

Graph Algorithms

- Search
  - Depth-first search
  - Breadth-first search
- Shortest paths
  - Dijkstra's algorithm
- Minimum spanning trees
  - Prim's algorithm
  - Kruskal's algorithm