A Survey of Graph Theory and Applications in Neo4J

Baltimore Washington

Graph Database Meetup

About me

Geoff Moes

moesg@verizon.net

@elegantcoding

www.elegantcoding.com

Software developer and math enthusiast.

No hard questions, please!

What to Expect from this Talk

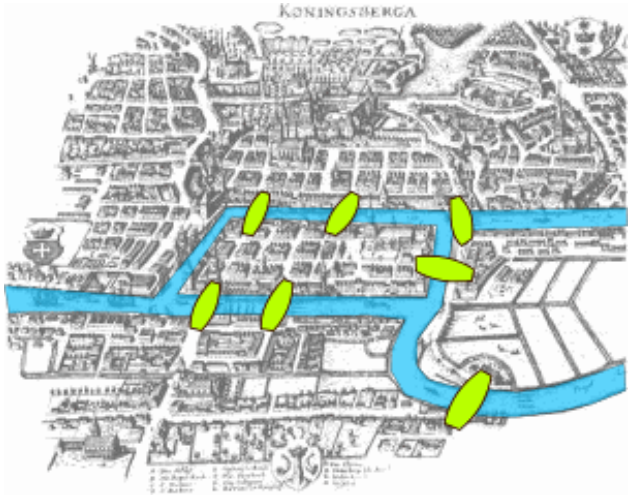
- Look at the ubiquity of graphs
- General overview of what graphs are and how they work
- Explore ideas about the language, structure, and patterns of graphs
- The structure of the unstructured
- Goal is not to teach these concepts but to show them
- The tip of the iceberg
- Lots of pictures

Why Learn Graph Theory

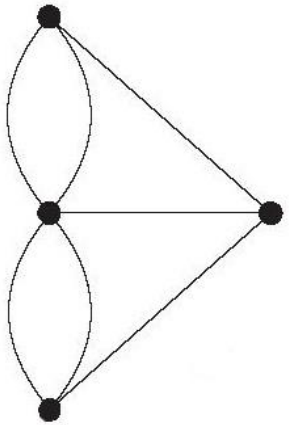
- Many real world and abstract concepts exhibit relationships and connectivity. Graphs are really are everywhere.
- We live in a connected world, increasingly ever more so.
- Graph Theory is very intuitive and its ubiquity makes it very natural.
- You probably never learned in school so the educational system didn’t ruin it for you.
- It is about connectedness and it connects to a large number of areas of math. Gateway drug for other maths.
- It is central to many areas of CS

A Brief History of Graph Theory

Seven Bridges of Königsberg (1735) is obligatory but is also a watershed event for the development of graph theory and topology (rubber sheet geometry).

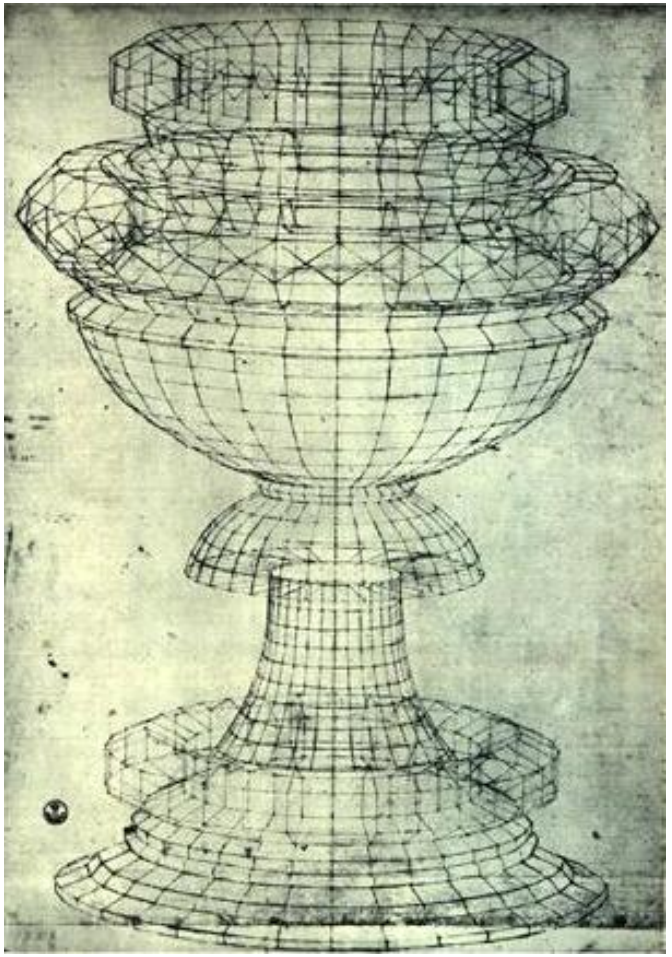


Euler recognized this problem as a geometric problem where distance didn't matter. It is about connectedness.



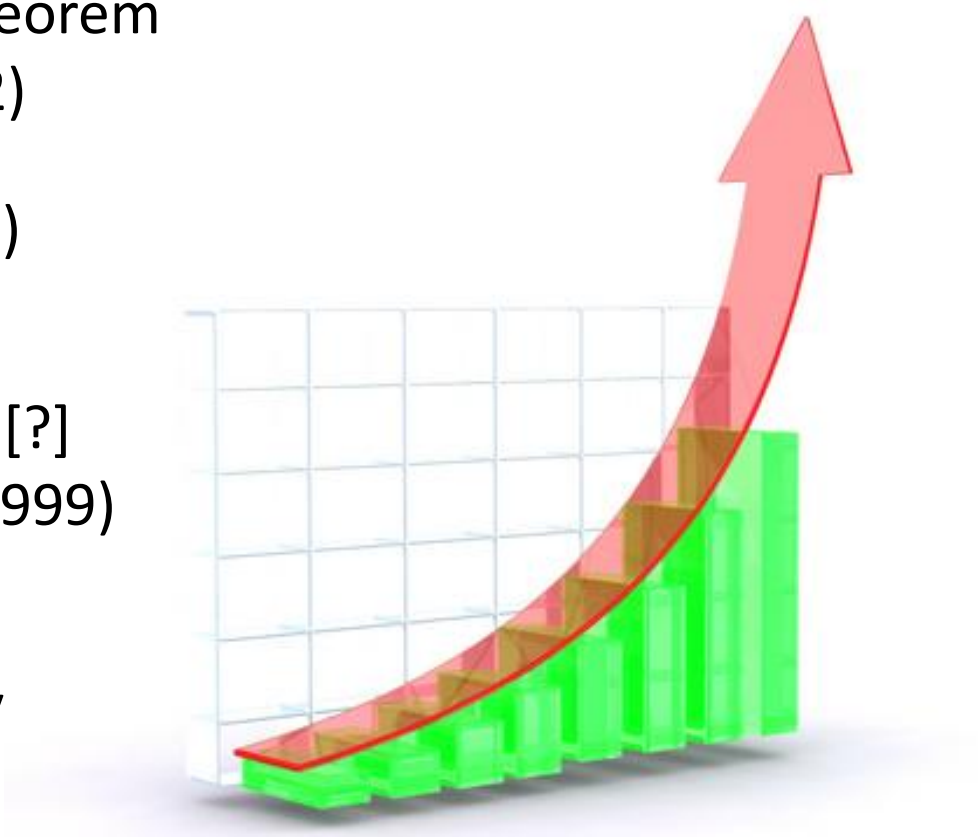
A Brief History of Graph Theory
19th Century

- Gustav Kirchhoff (1824-87) Trees in Electric Circuits
- Thomas Kirkman (1806-95), William Rowan Hamilton (1805-65) Hamiltonian cycles in polyhedra
- Arthur Cayley (1821-95), James J. Sylvester(1806-97), George Polya(1887-1985), and others use graphs to enumerate chemical molecules
- Francis Guthrie (1831-99), Augustus De Morgan (1806-71) Four Colour Problem (1852)



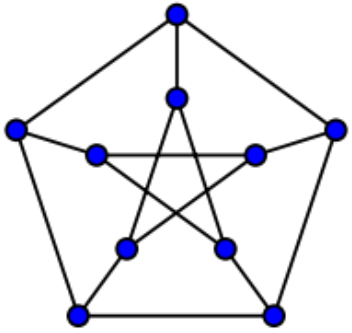
A Brief History of Graph Theory
20th Century

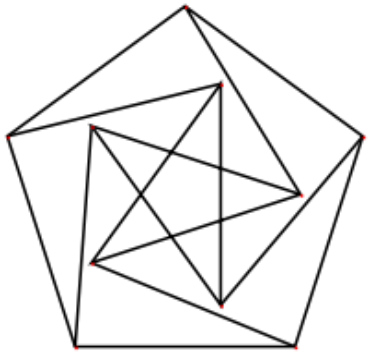
- Kazimierz Kuratowski (1896-1980) Kuratowski's Theorem
- W. T. Tutte (1917-2002) Tutte Polynomial
- Erdős and Renyi (1959) Random Graphs
- Watts and Strogatz (1998/99) Alpha/Beta [?]
- Albert and Barabasi (1999) Power Laws
- Fan Chung [Graham] - Spectral Graph Theory
- Many more

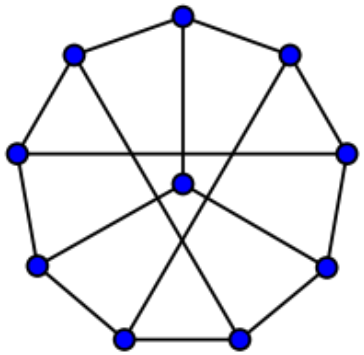


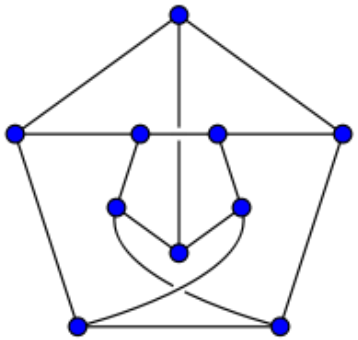
What are Graphs?

- Graph theory concerns the relationship among lines and points.
- A graph consists of some points and some lines between them.
- No attention is paid to the position of points and the length of the lines.
- Vertices are adjacent if there is an edge connecting them.
- Relationships

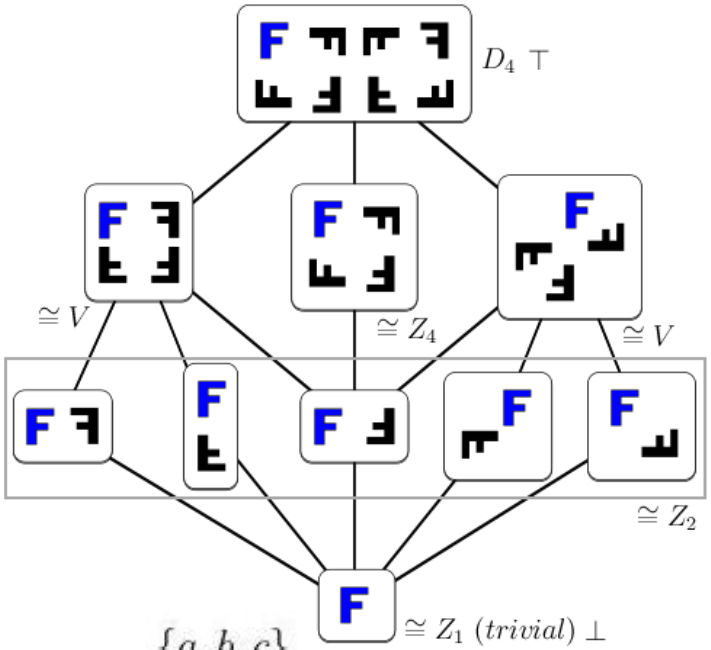


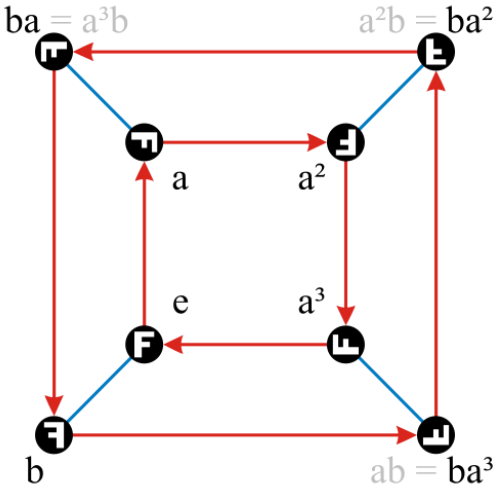


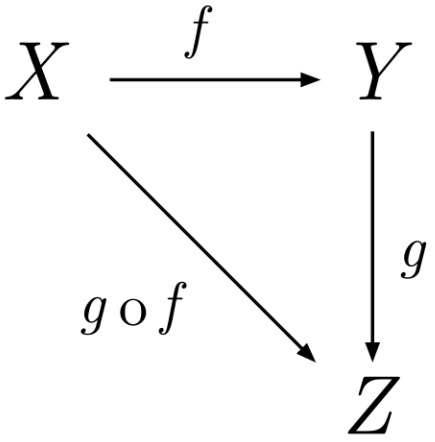


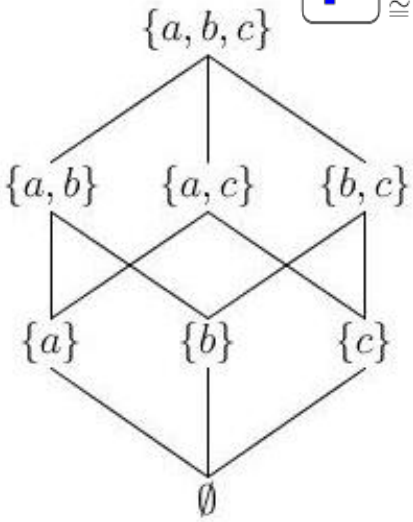


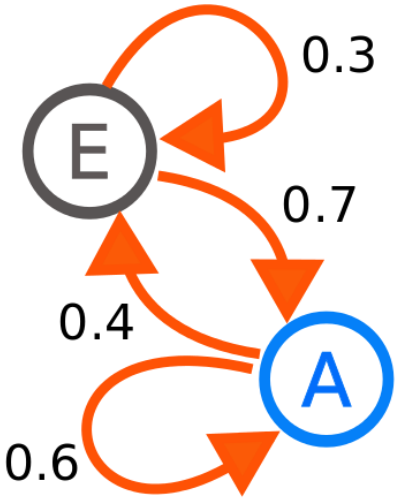
Graphs are Everywhere: Math

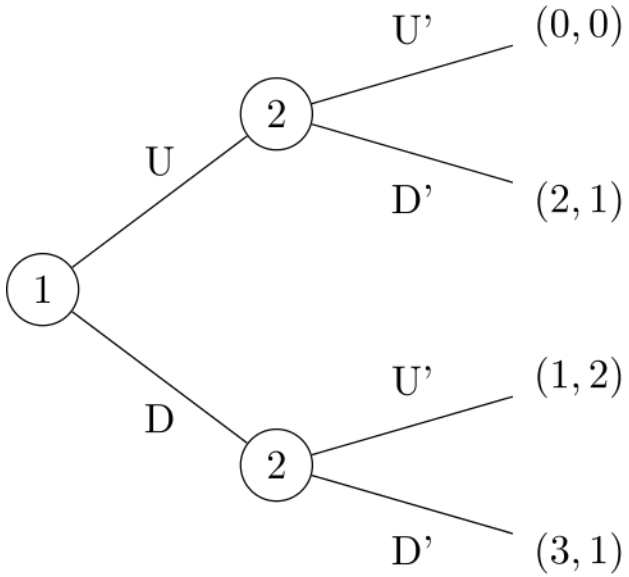




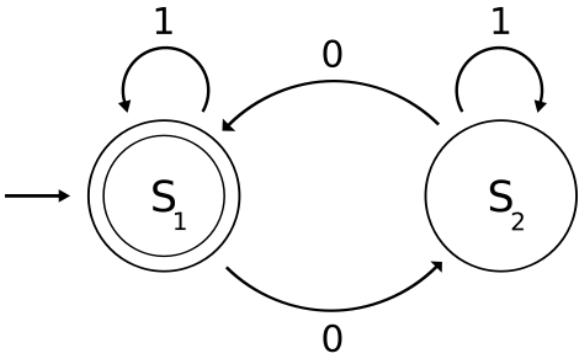


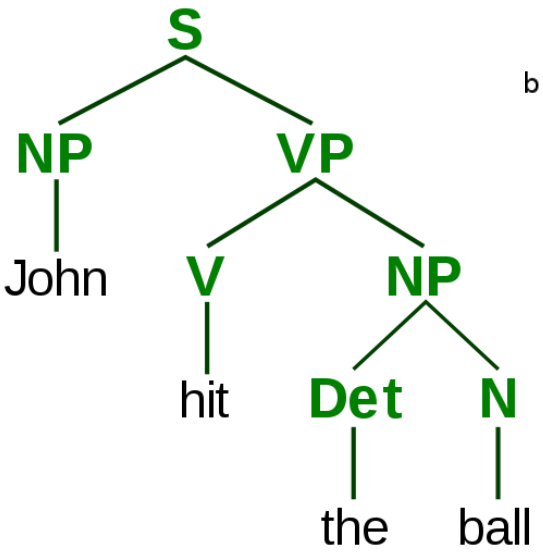


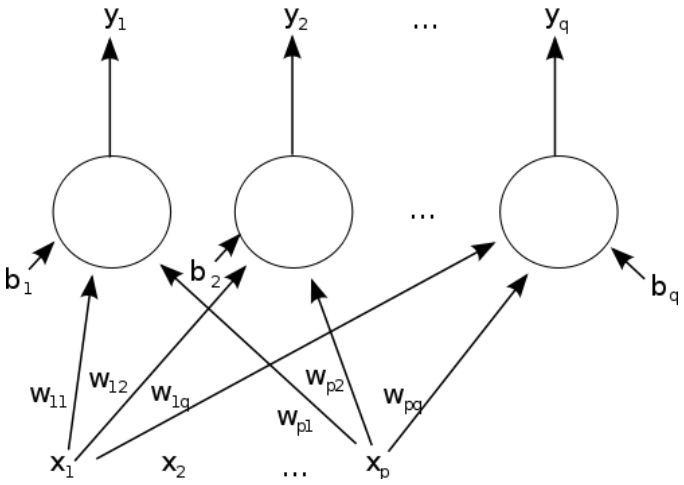


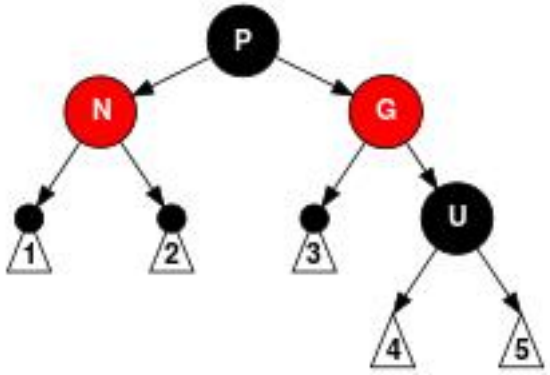


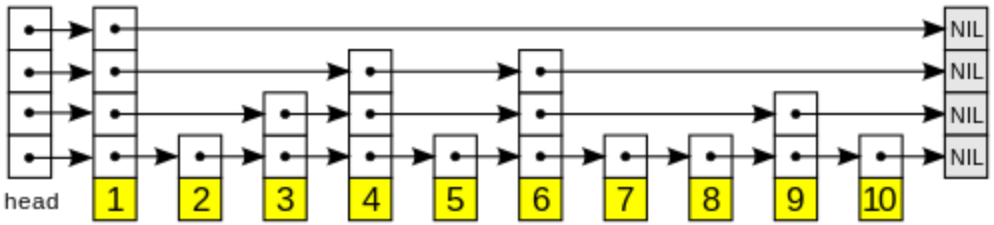
Graphs are Everywhere: CS

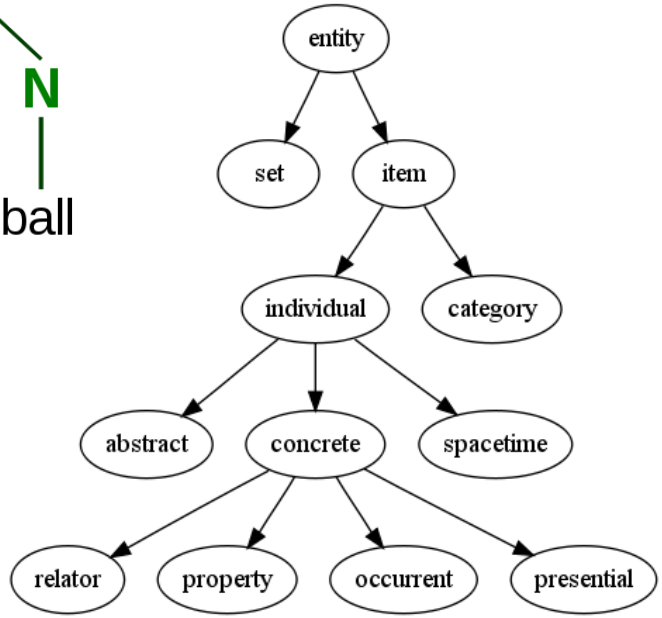




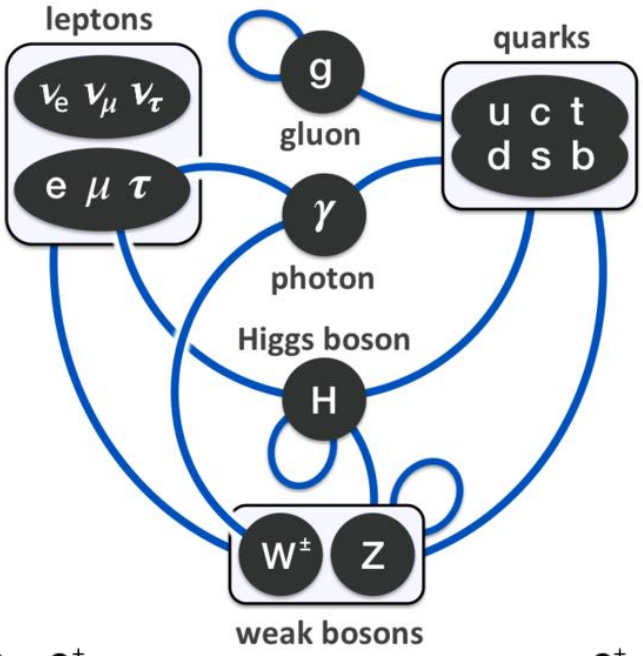


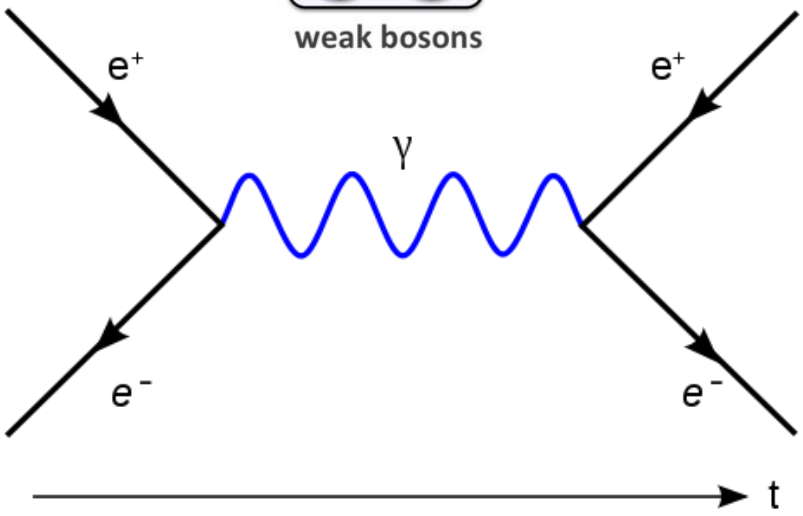


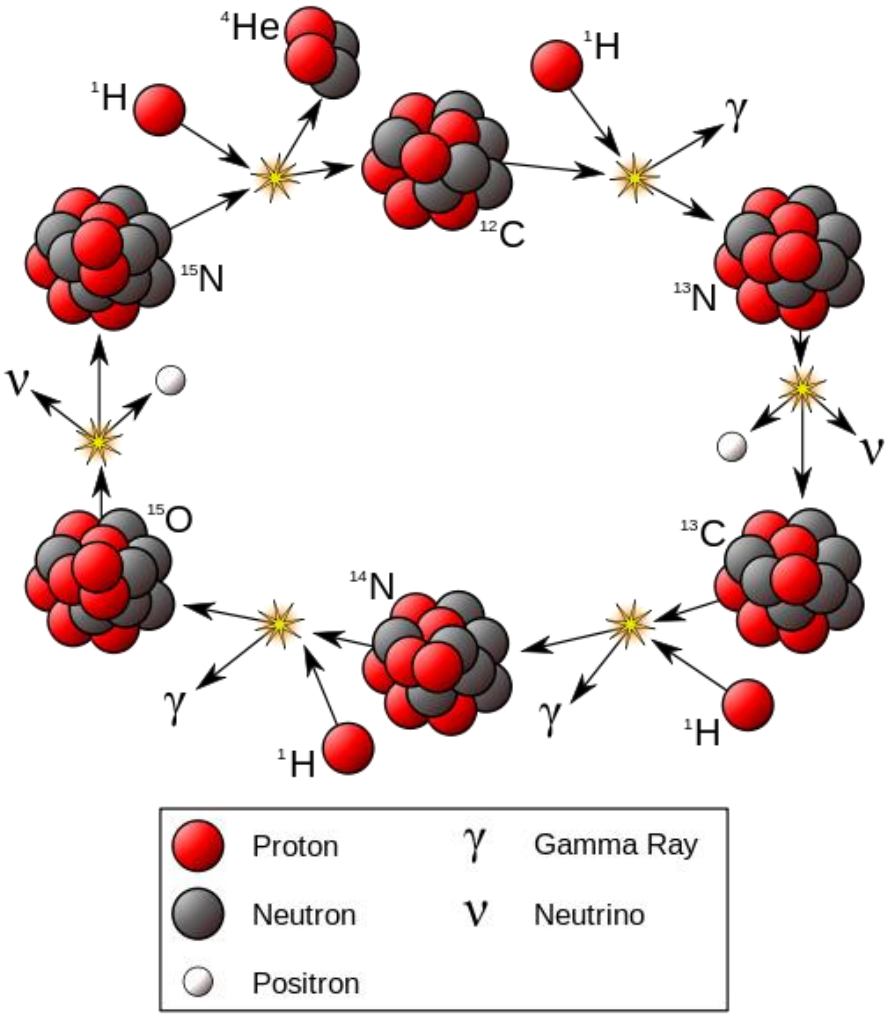




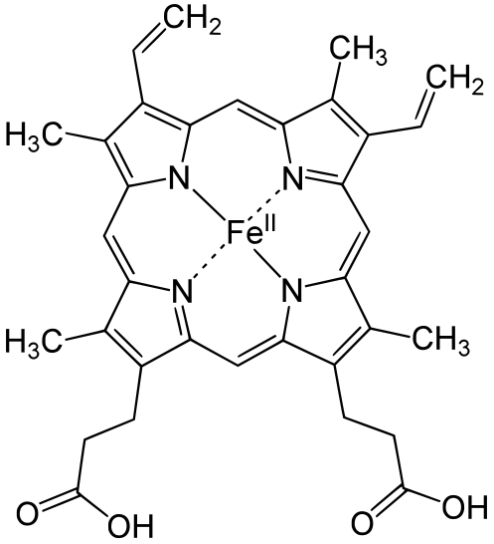
Graphs are Everywhere: Physics

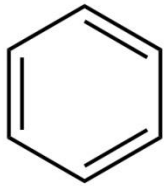


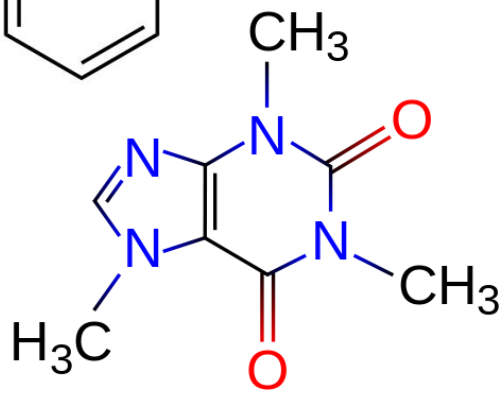


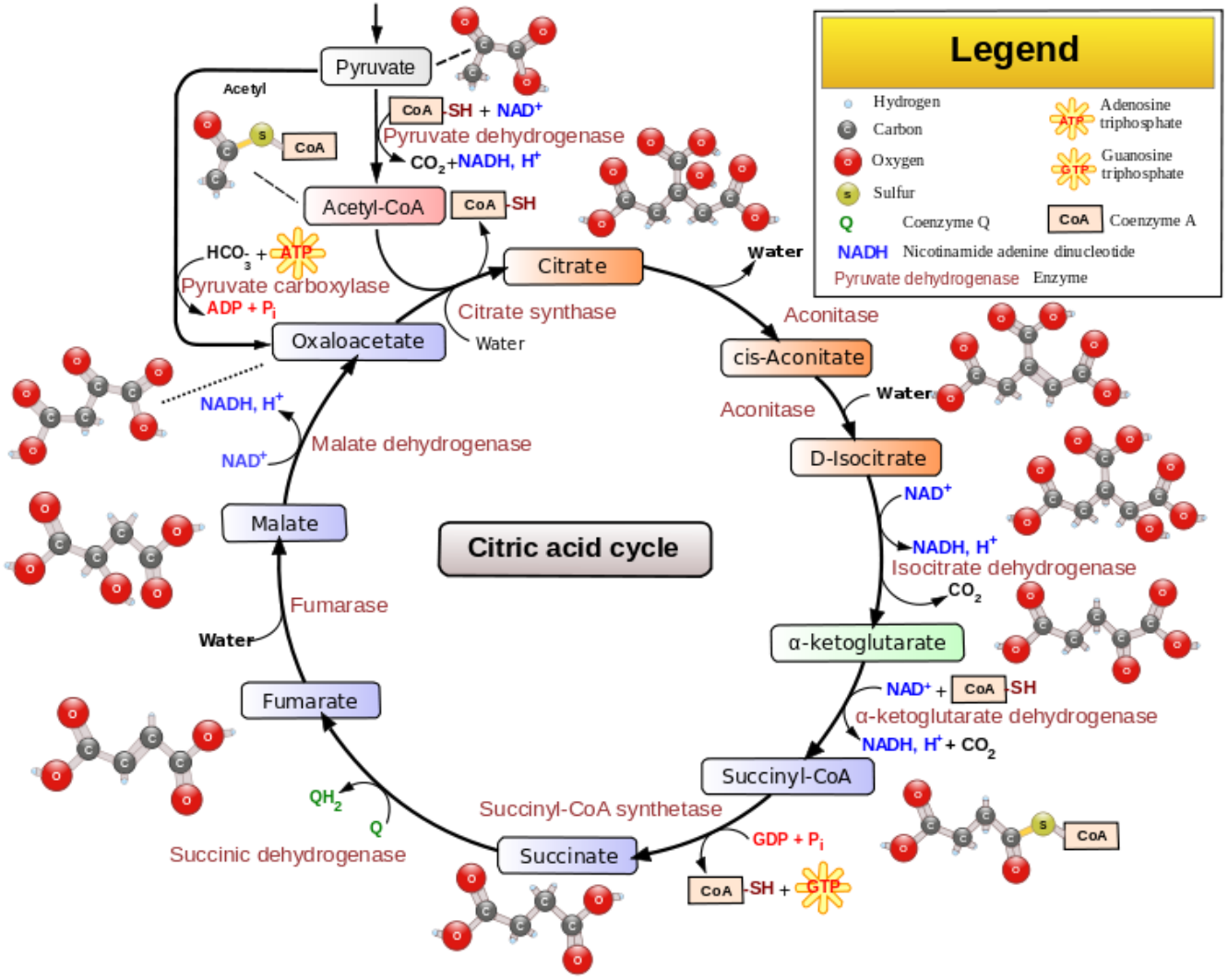


Graphs are Everywhere: [Bio]Chem

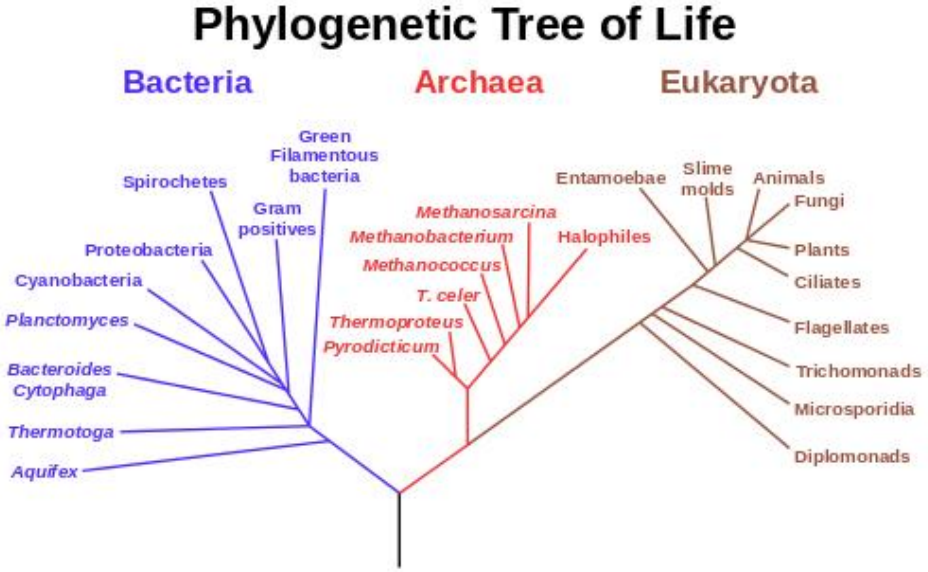


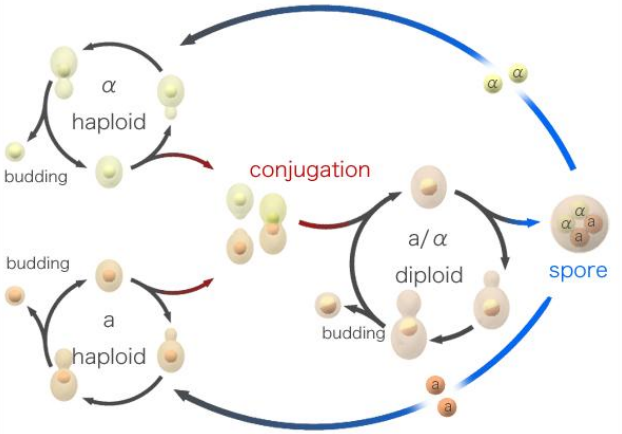


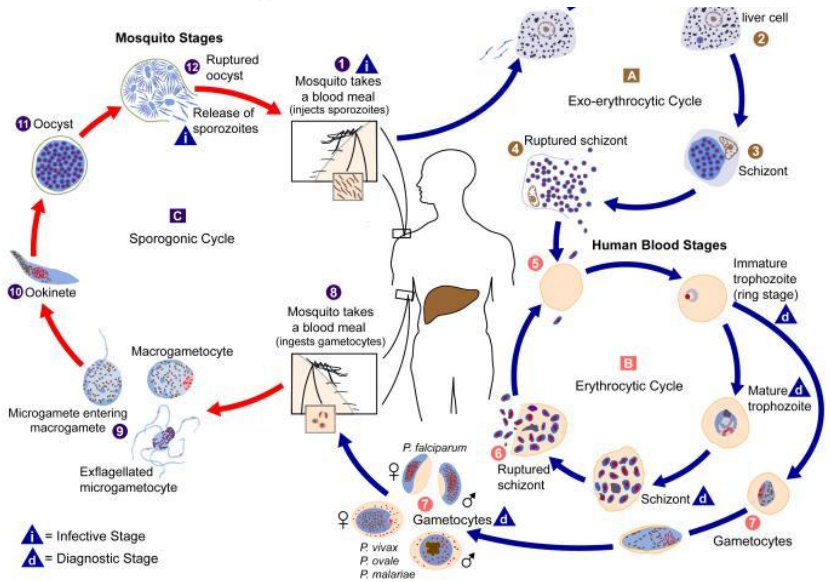


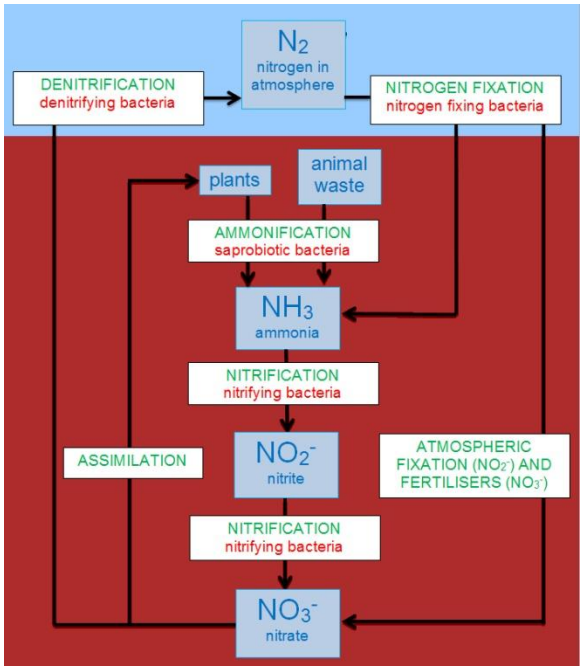


Graphs are Everywhere: Biology

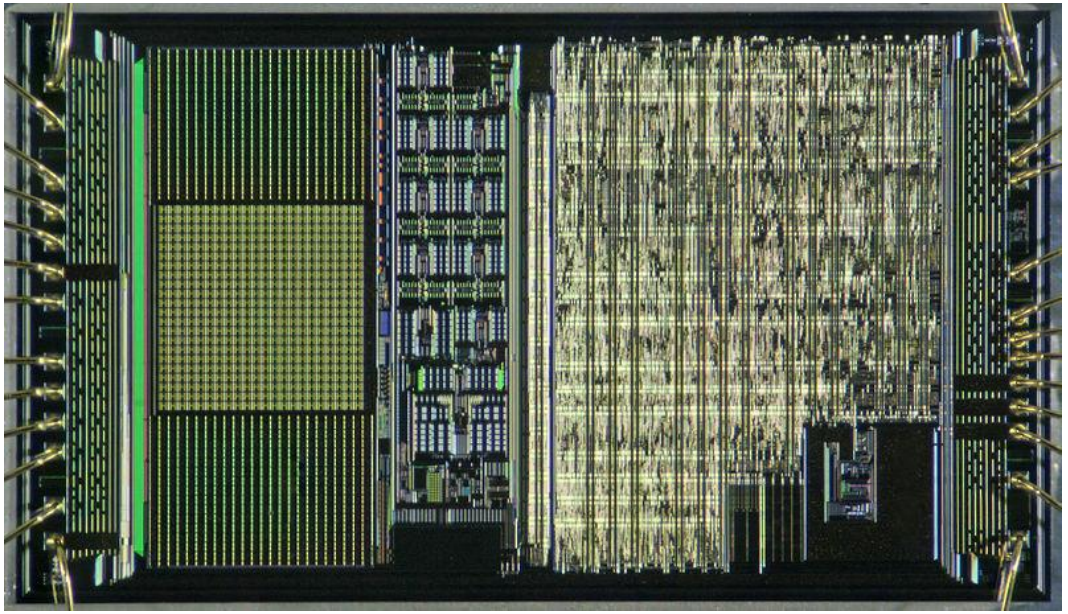


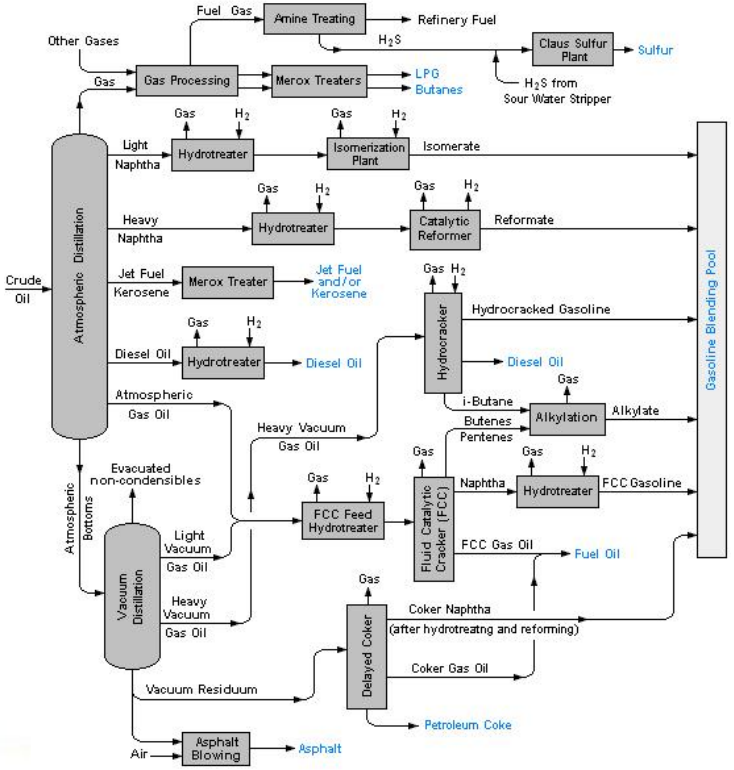


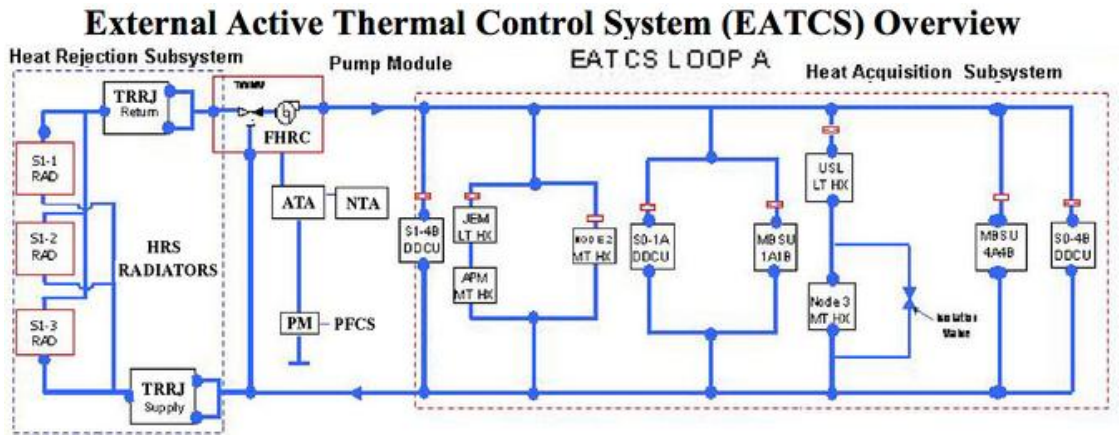


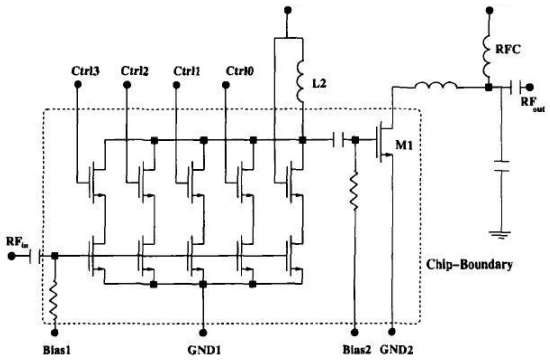


Graphs are Everywhere: Engineering

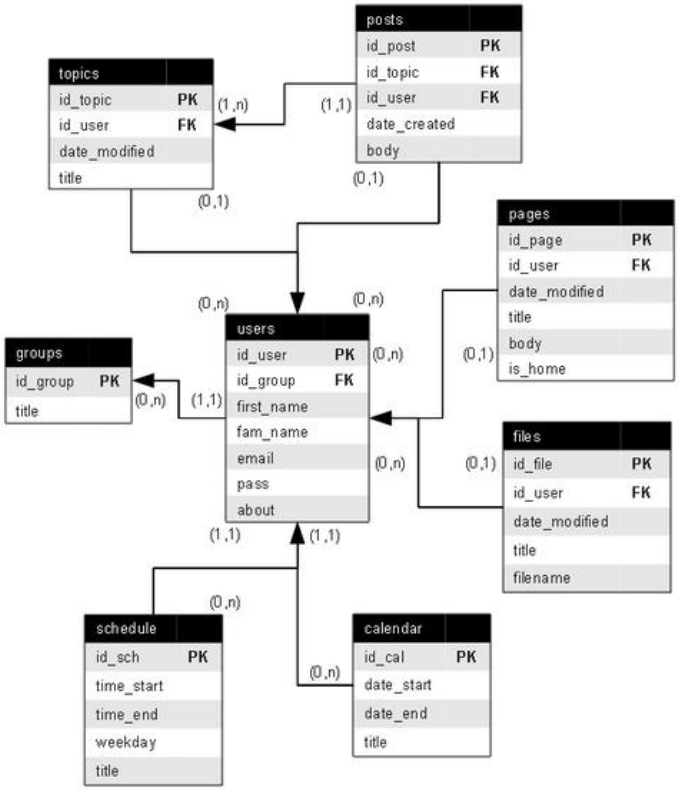


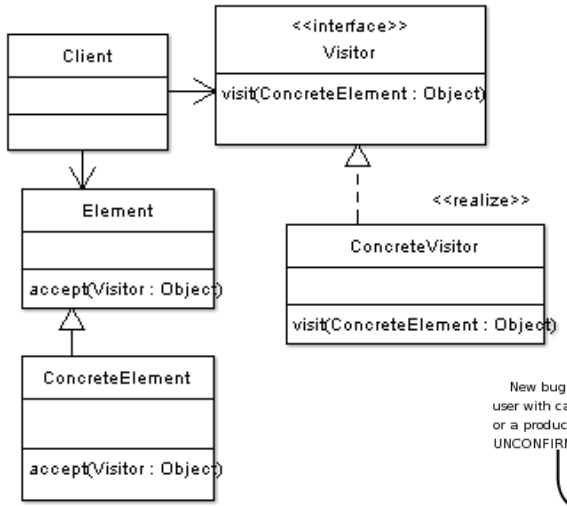


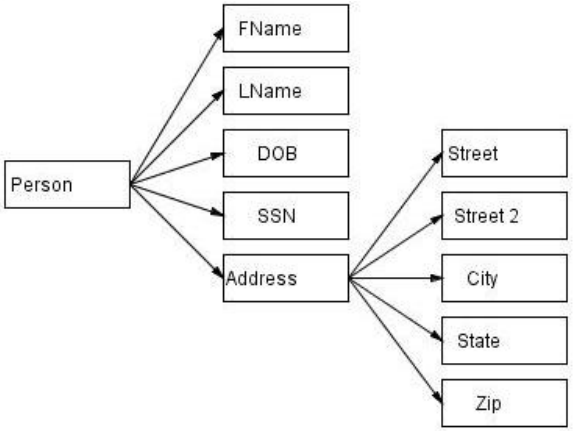


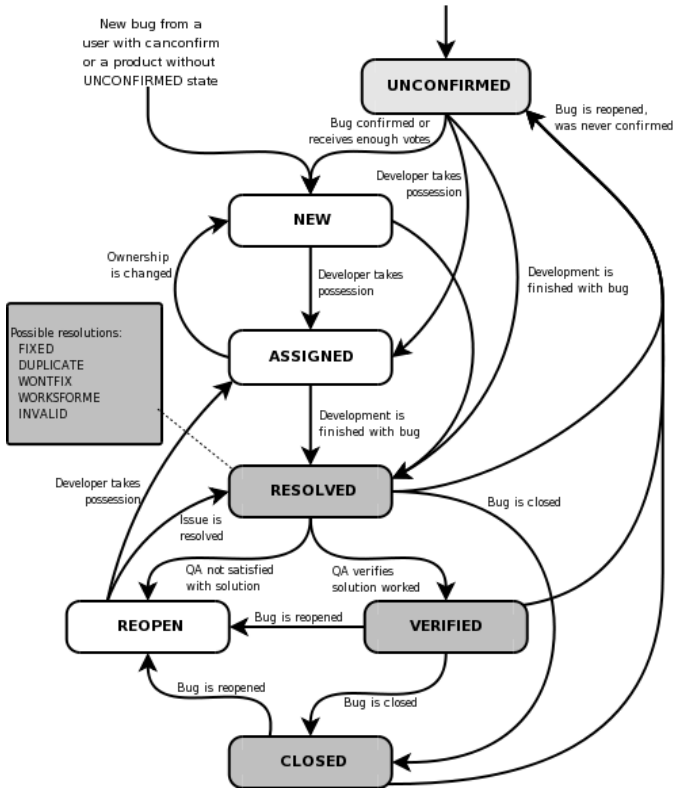


Graphs are Everywhere: Software



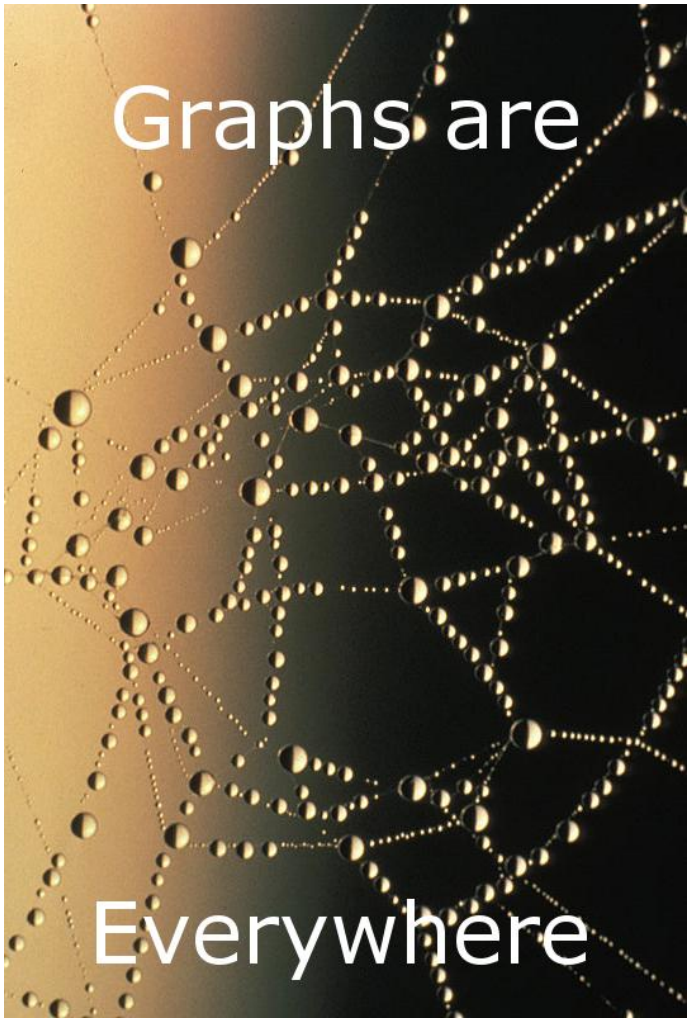






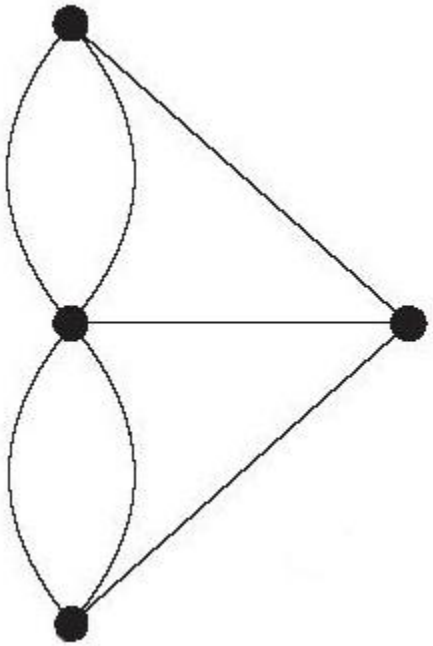
“Categorization” or “Kinds of” graphs

- Some basic ideas to describe graphs
- Classification, Meta-structural (basic structural qualities)
- Some of these have important distinctions for matrices
- Finite vs. Infinite, we just care about finite here.



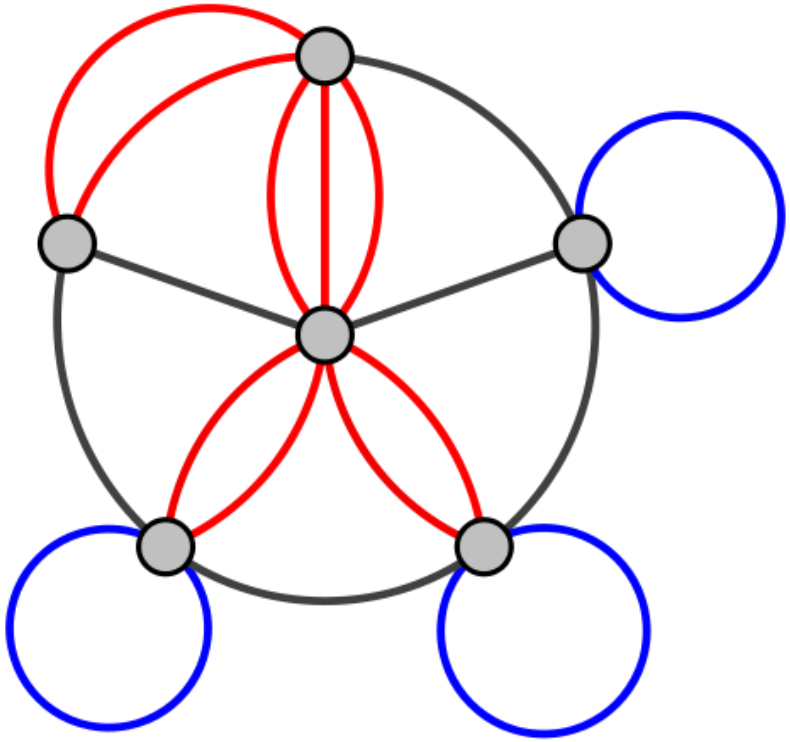
“Categorization” or “Kinds of” graphs

- **Multigraphs** allow multiple edges between two vertices
- AKA **Quiver**



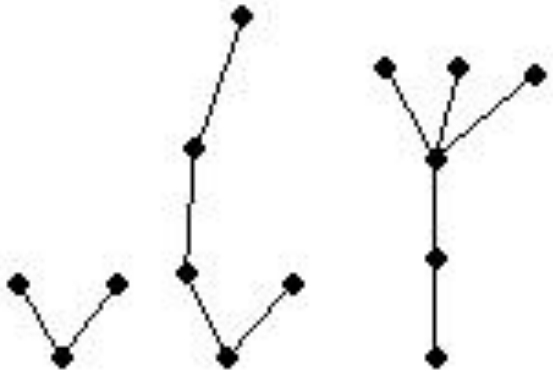
“Categorization” or “Kinds of” graphs

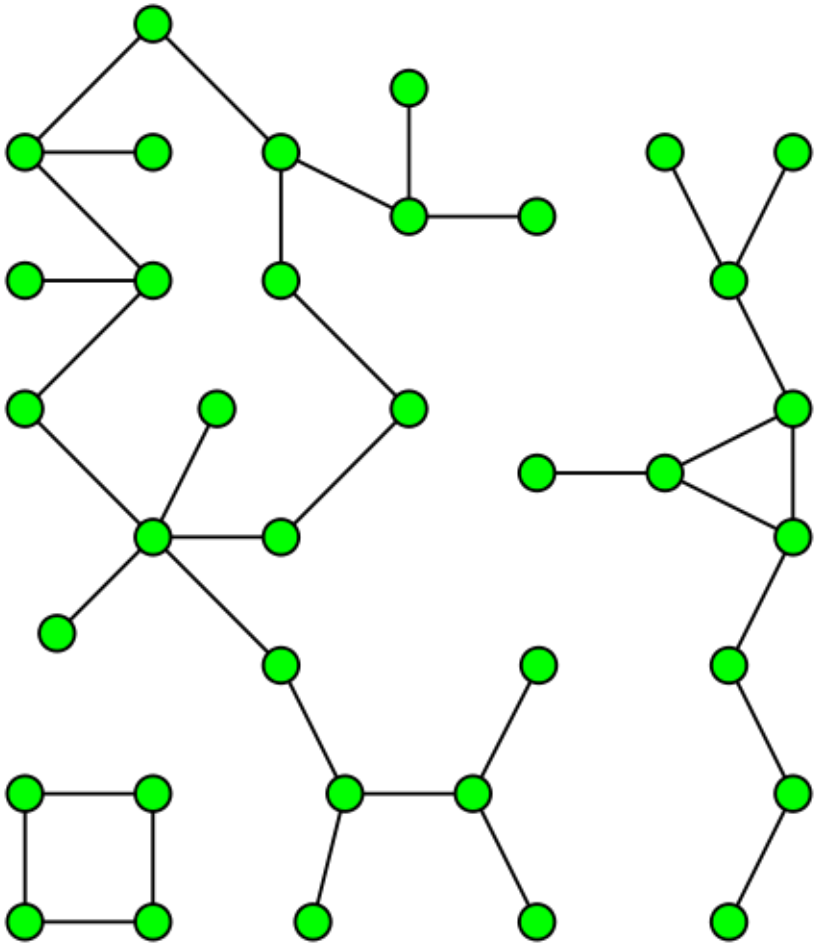
- **Loops** are a connection from a vertex to itself
- Reflexive Relation
- Loops can appear in both simple and multigraphs
- **Multiple Loops** can exist



“Categorization” or “Kinds of” graphs

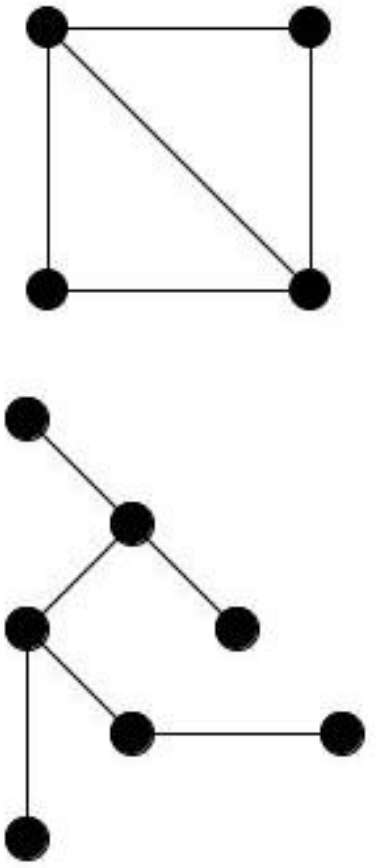
- A single graph can consist of multiple C**onnected Components Pseudoforrest**.
- **Trees** in a forest.





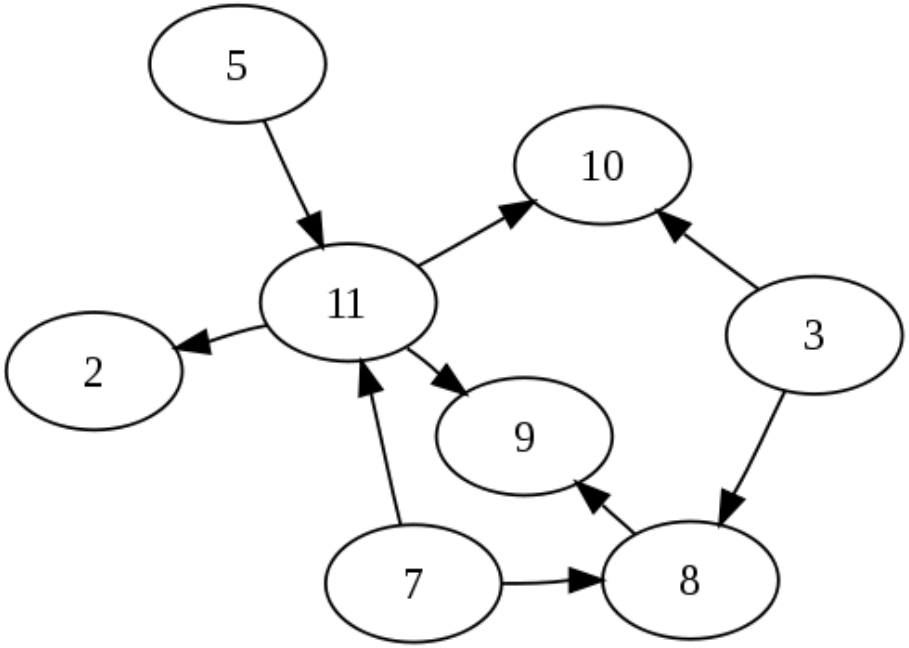
“Categorization” or “Kinds of” graphs

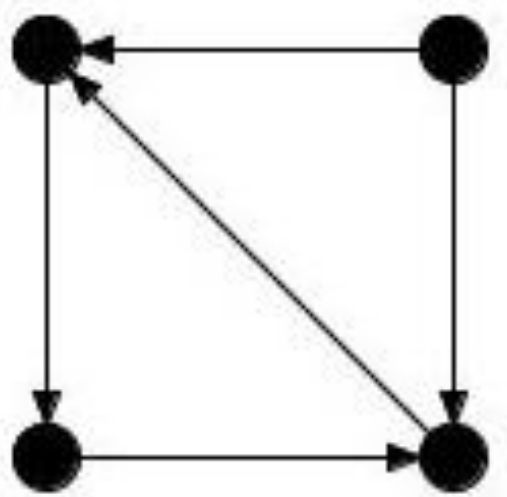
- Simple (Simplical) Graphs or just regular old Graphs
- No Loops
- Single edges between vertices
- Undirected edges



“Categorization” or “Kinds of” graphs

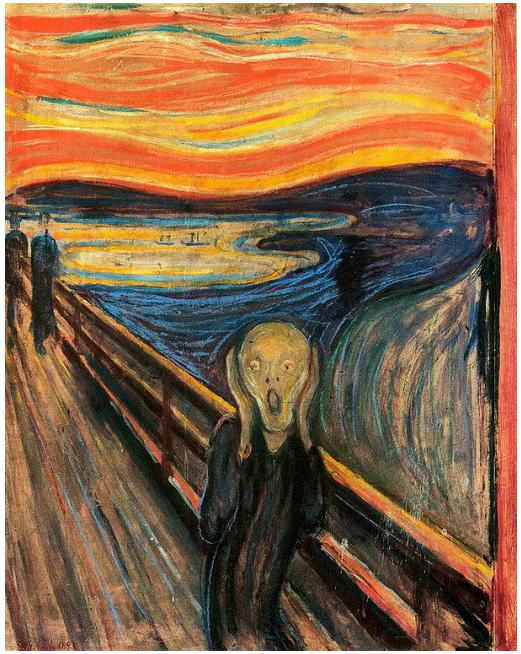
- Directed Graphs (Digraphs)
- Directed Acyclic Graphs (DAGs)

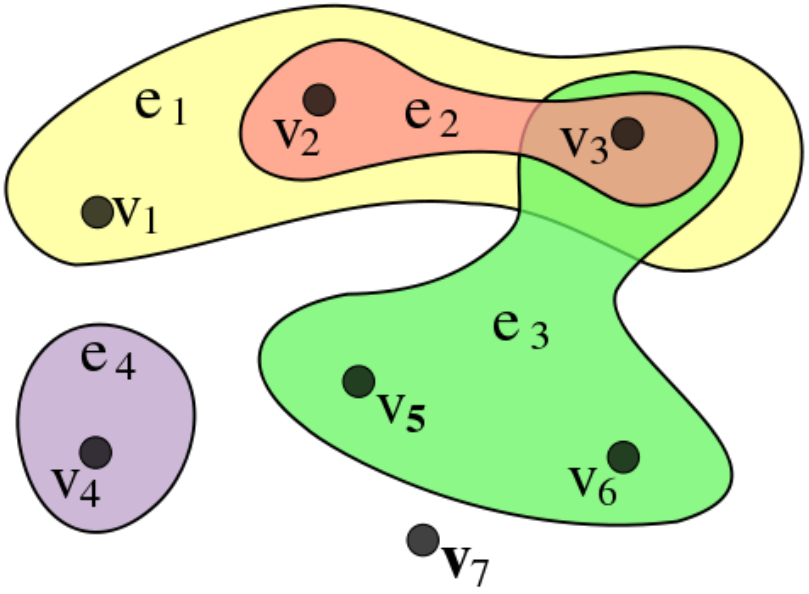




“Categorization” or “Kinds of” graphs

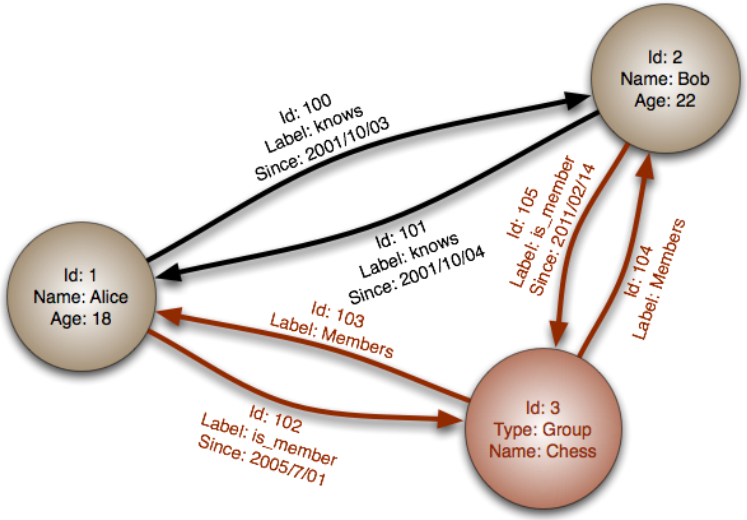
- Hypergraphs
- Oh No!

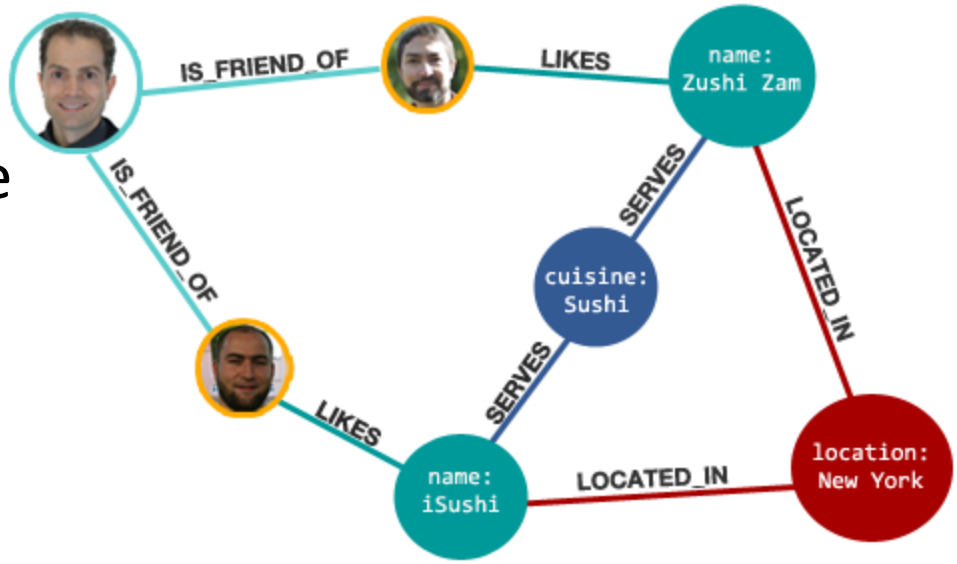




“Categorization” or “Kinds of” graphs

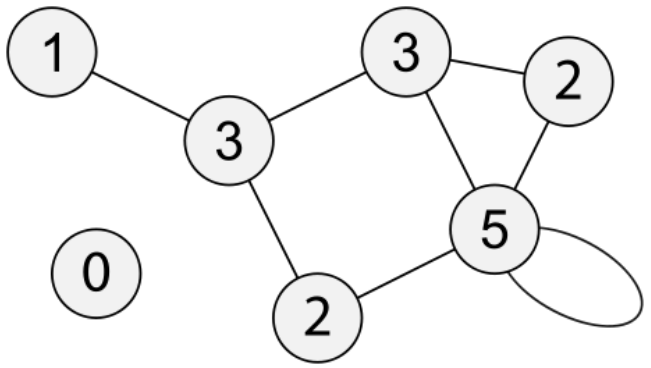
- What kind of graph is Neo4J?
- Directed Multigraph with Multiple Loops (Multi-Loop Quiver)
- Cypher allows edges to be undirected in queries.
- Undirected edges are not first class [stored] objects





A Couple of Basic Ideas

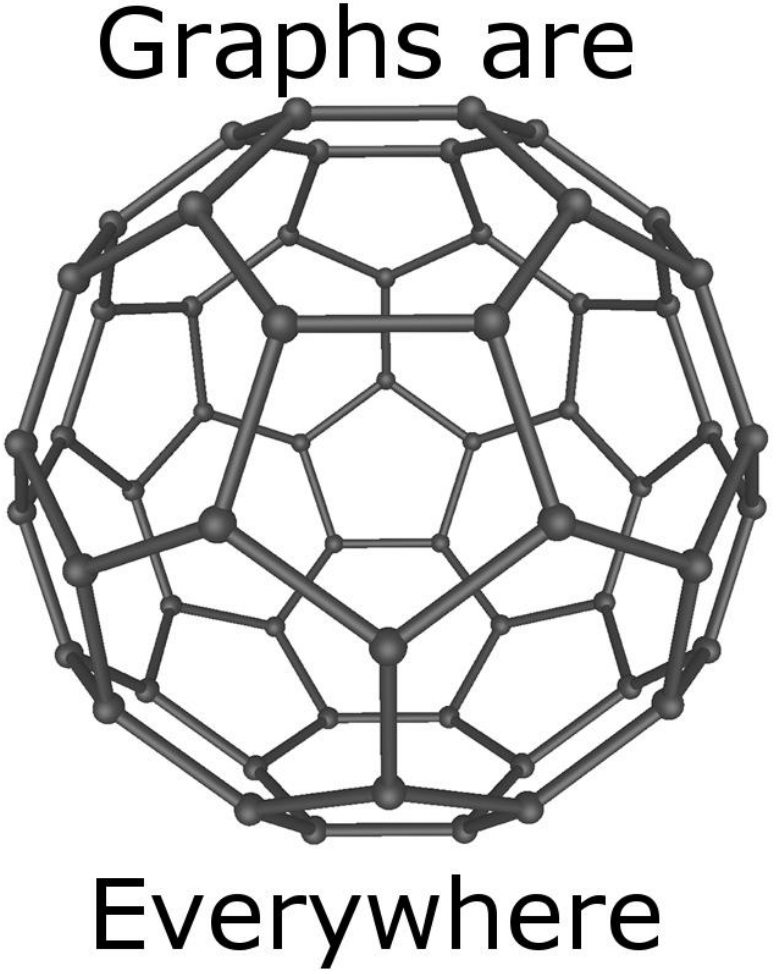
- The degree of a vertex is the number of edges that are connected to it, denoted d(v). In a directed graph a vertex can have both an inbound and outbound degree.
- Handshaking Lemma: Every finite undirected graph has an even number of vertices with odd degree.



$$\sum_{v \in G} d(v) = 2 \cdot |E_G|$$

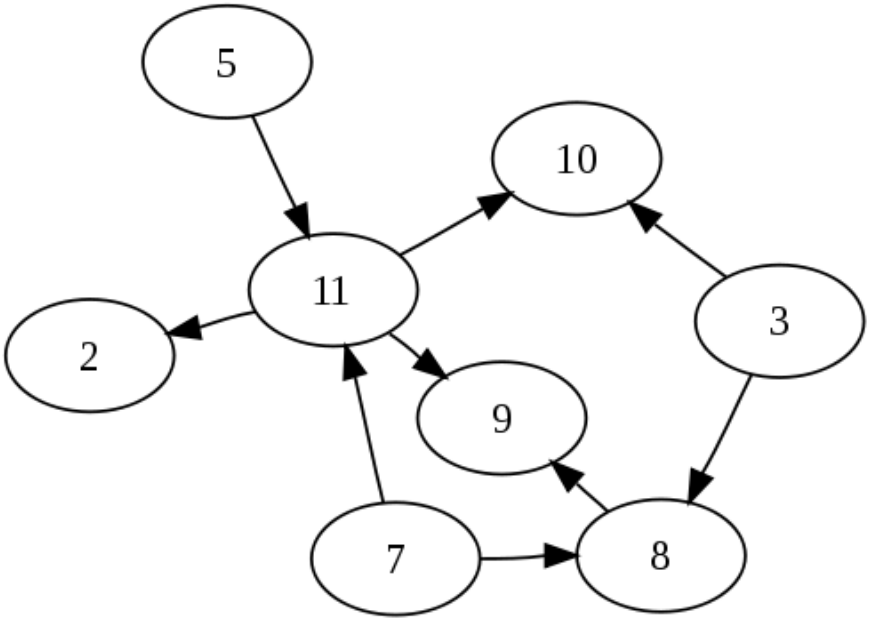
Special or Structural Types of Graphs

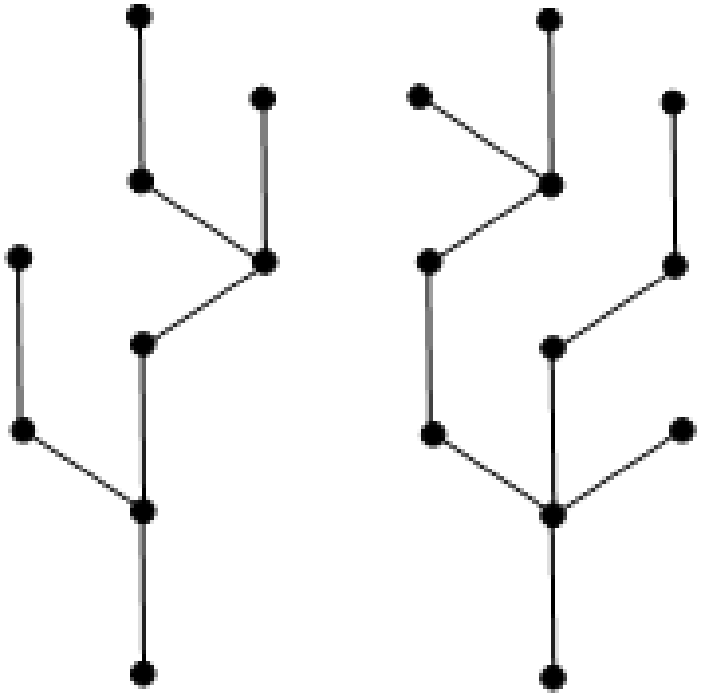
- Mostly Simple Graphs.
- There are a number of classifications of graphs based on properties and patterns of graphs.



Special or Structural Types of Graphs

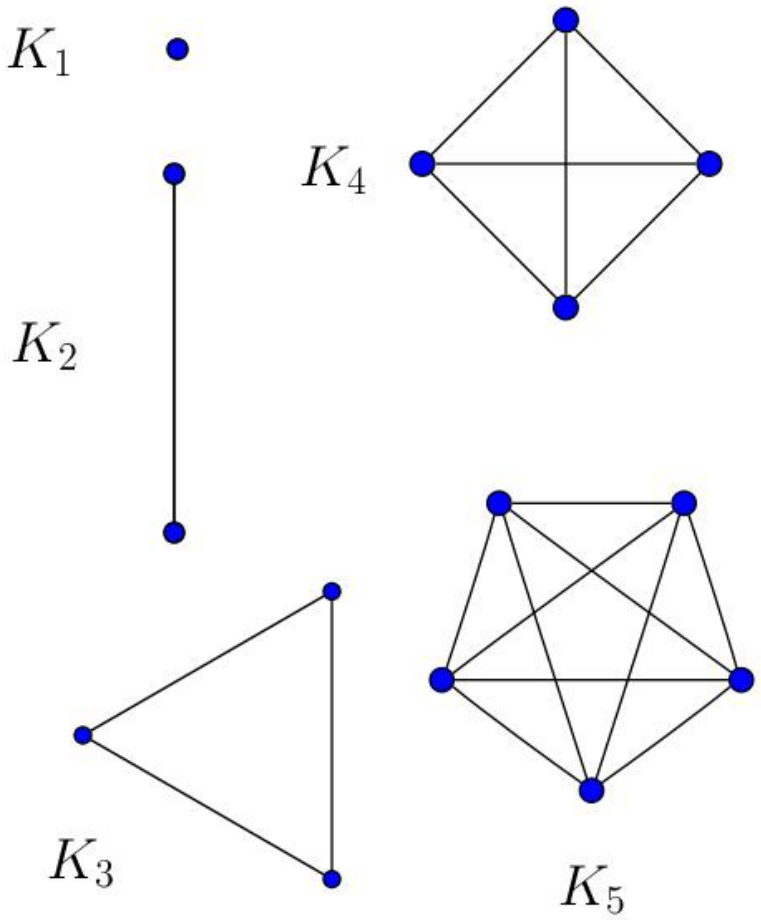
- Acyclic graphs
- Graphs without cycles
- DAGs and Trees





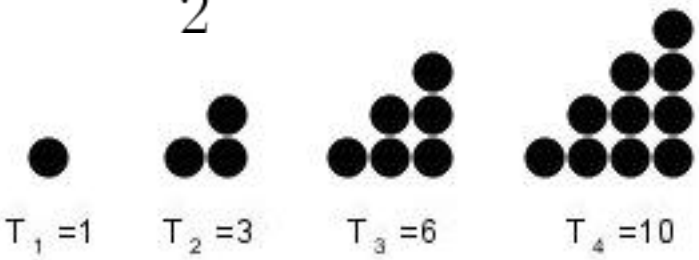
Special or Structural Types of Graphs

- Complete graphs
- All vertices are connected by one edge
- Vertices n-1 degree
- Regular
- Number of edges n-1 Triangular number



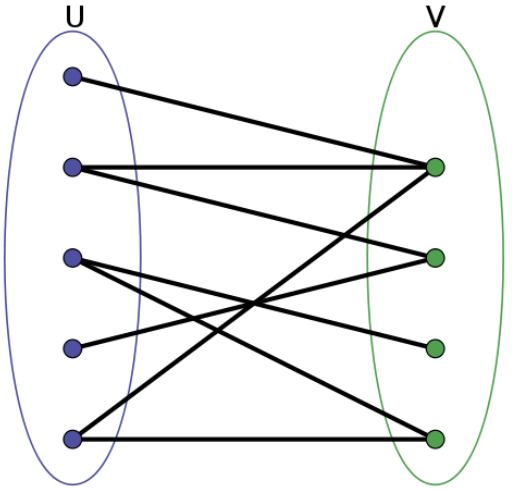
$$T_{n-1} = \frac{n^2 - n}{2}$$

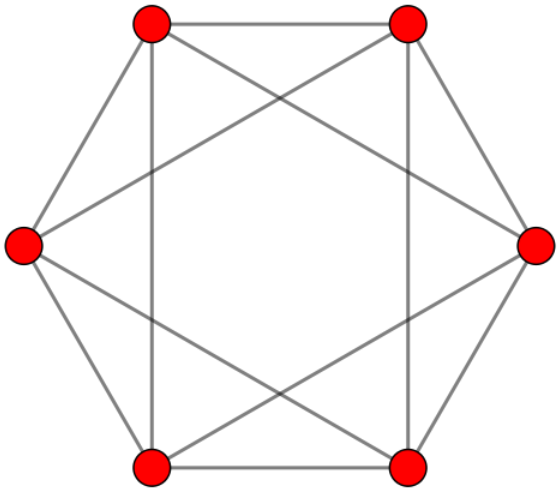
$$T_n = \sum_{k=1}^{n} k$$

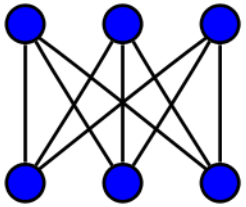


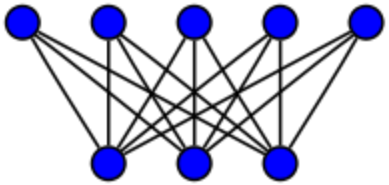
Special or Structural Types of Graphs

- Bipartite, Complete Bipartite, Multipartite, Complete Multipartite
- Partition of vertices
- Complete: Utility Graph $K_{3,3}$, $K_{3,5}$, $K_{2,2,2}$
- All acyclic graphs are Bipartite



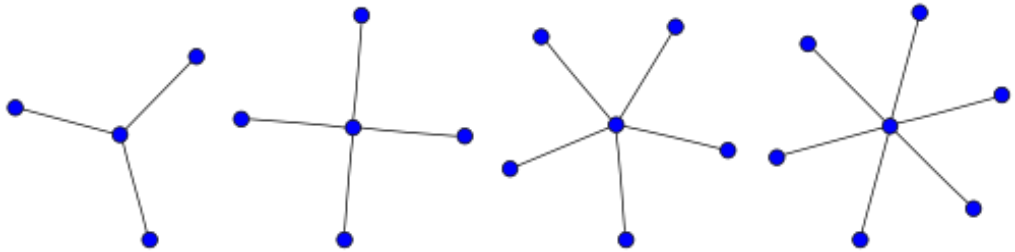


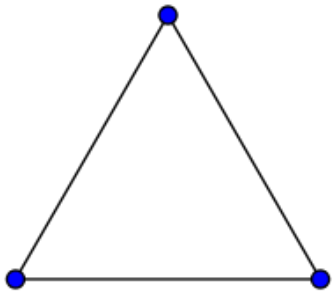


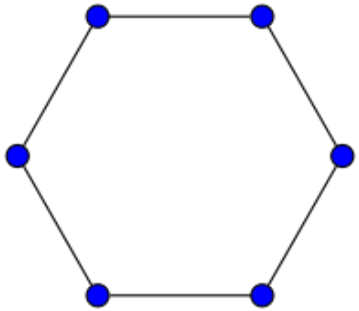


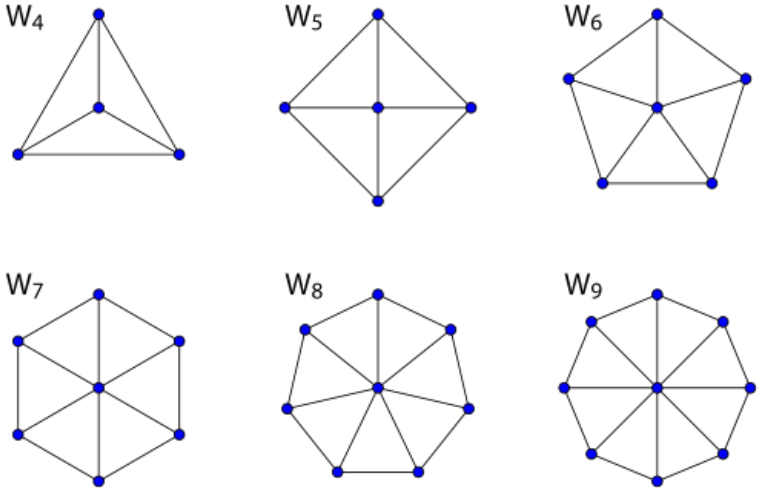
Special or Structural Types of Graphs

- Path Graphs P_n
- Star Graphs S_n ($K_{1,n}$)
- Wheel Graphs W_n,
- Cycle Graphs C_n - n edges and n vertices, 2-regular



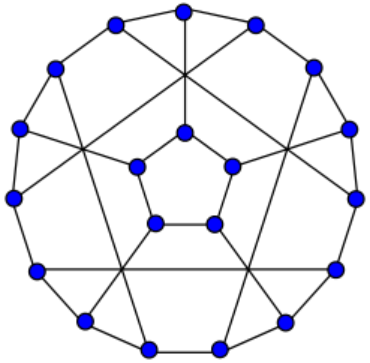


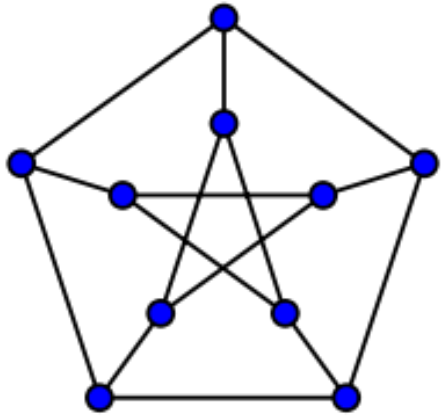


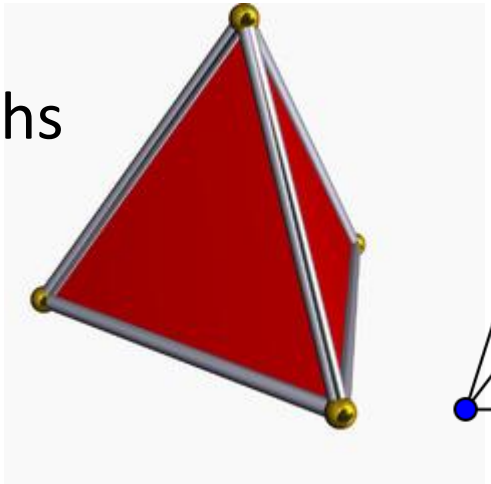


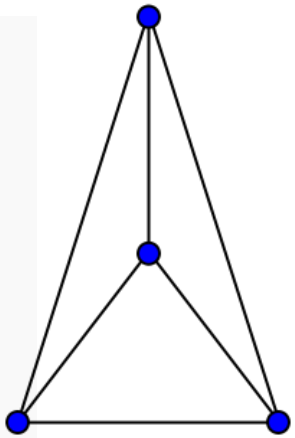
Special or Structural Types of Graphs

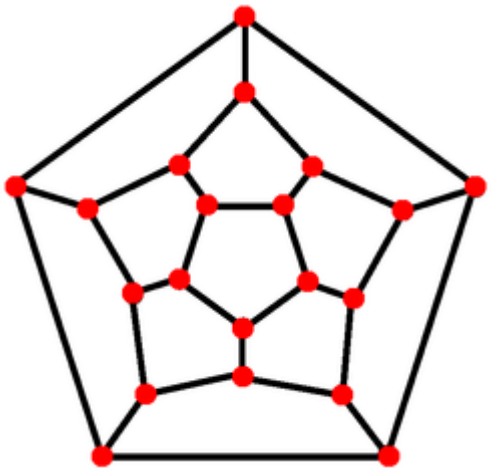
- Regular Graphs : Every vertex has the same degree.
- Snarks
- Petersen Graph, 3-Regular (Cubic)
- Polyhedral Graphs
- Random Graphs
- Dense Graphs
- Expander Graphs
- Many others











Graphs and Set Theory

- Graphs are defined in terms of sets
- A graph is an ordered pair G = (V, E) comprising a set V of vertices or nodes together with a set E of edges where

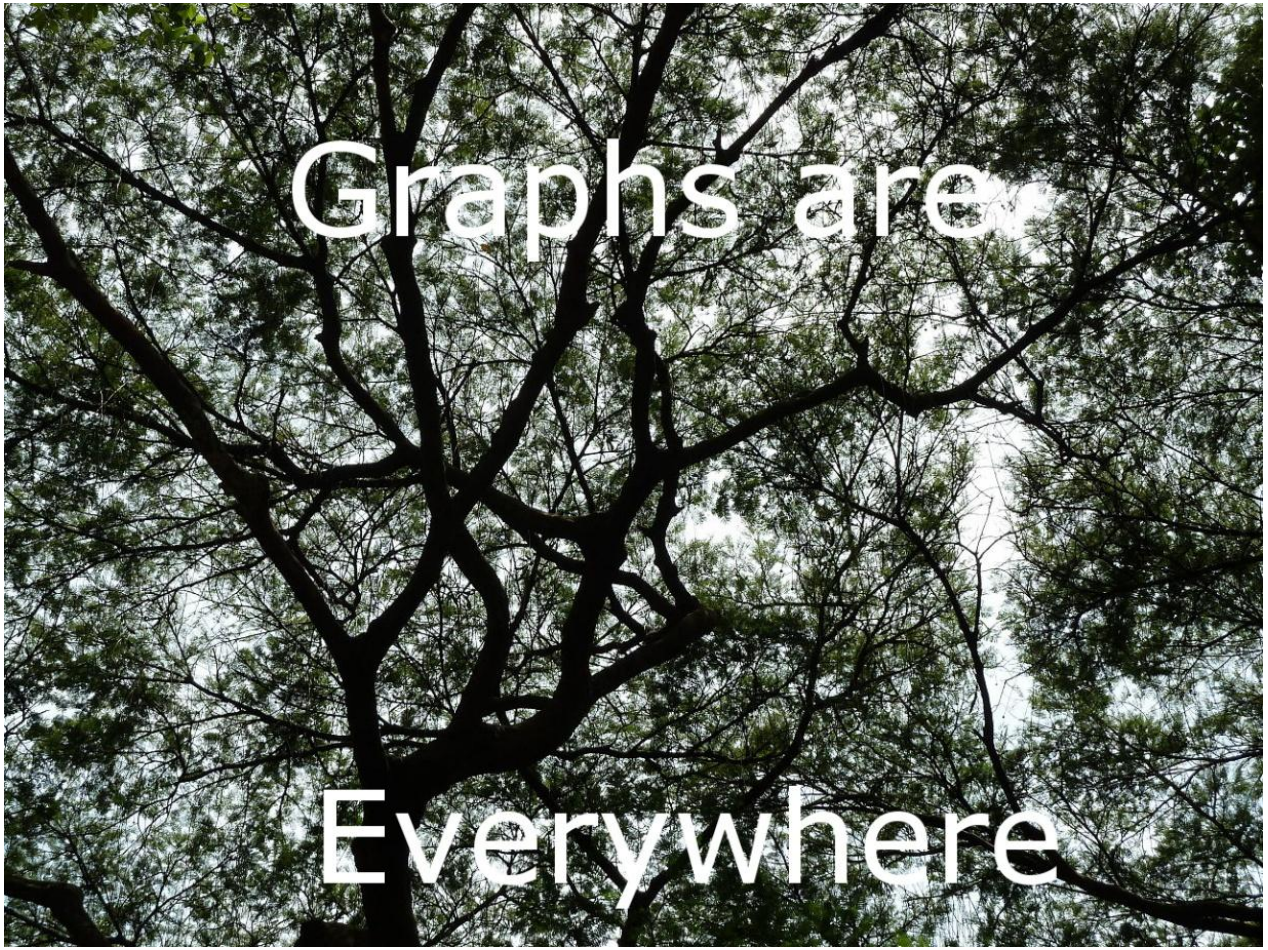
$$E \subseteq \{V \times V\}$$

- Graphs can combined under set operations like union, intersection, and subtraction (asymmetric difference)
- A multigraph would have a multiset (aka bag) of edges.

Graph Theory and Topology

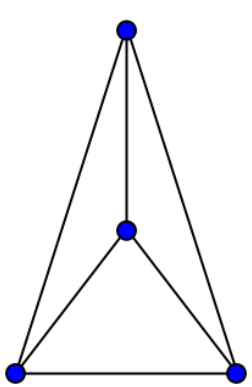
- Graph Theory is a sub-discipline of Topology (rubber sheet geometry)

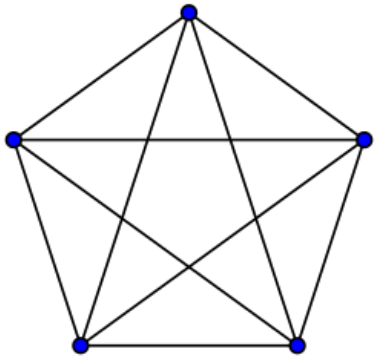
- Discipline of Topological Graph Theory includes studying spatial embedding of graphs

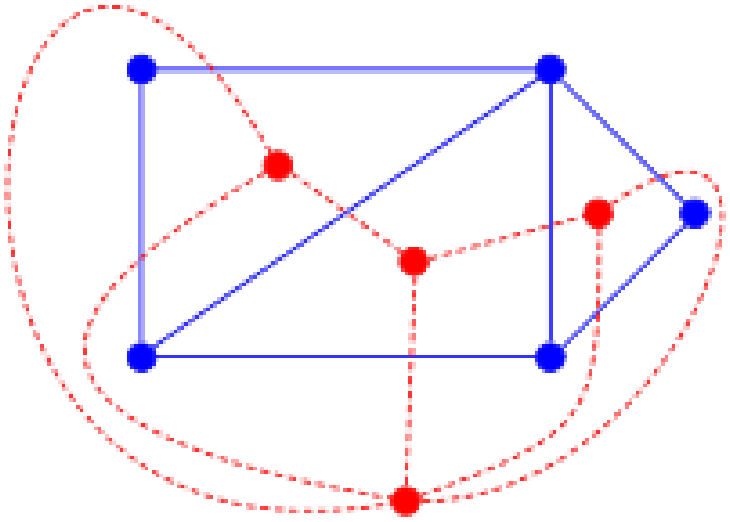


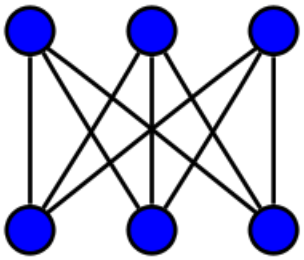
Graph Theory and Topology

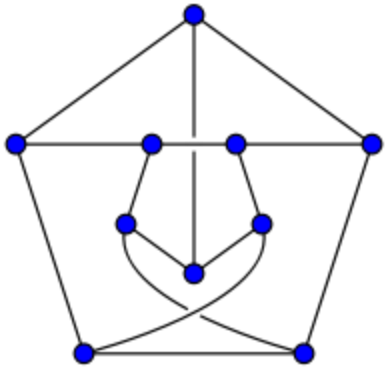
- Planarity
- Can be drawn in a plane such that edges do not cross.
- Non Planar include Utility, Petersen, K_5, etc.
- Crossing Number
- Dual Graph







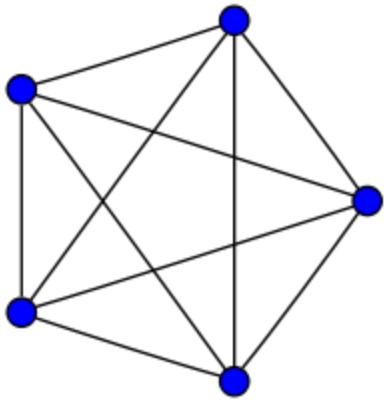


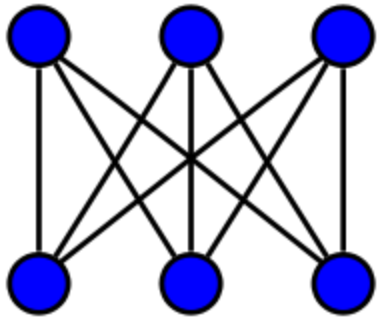


Graph Theory and Topology

- Kuratowski's Theorem

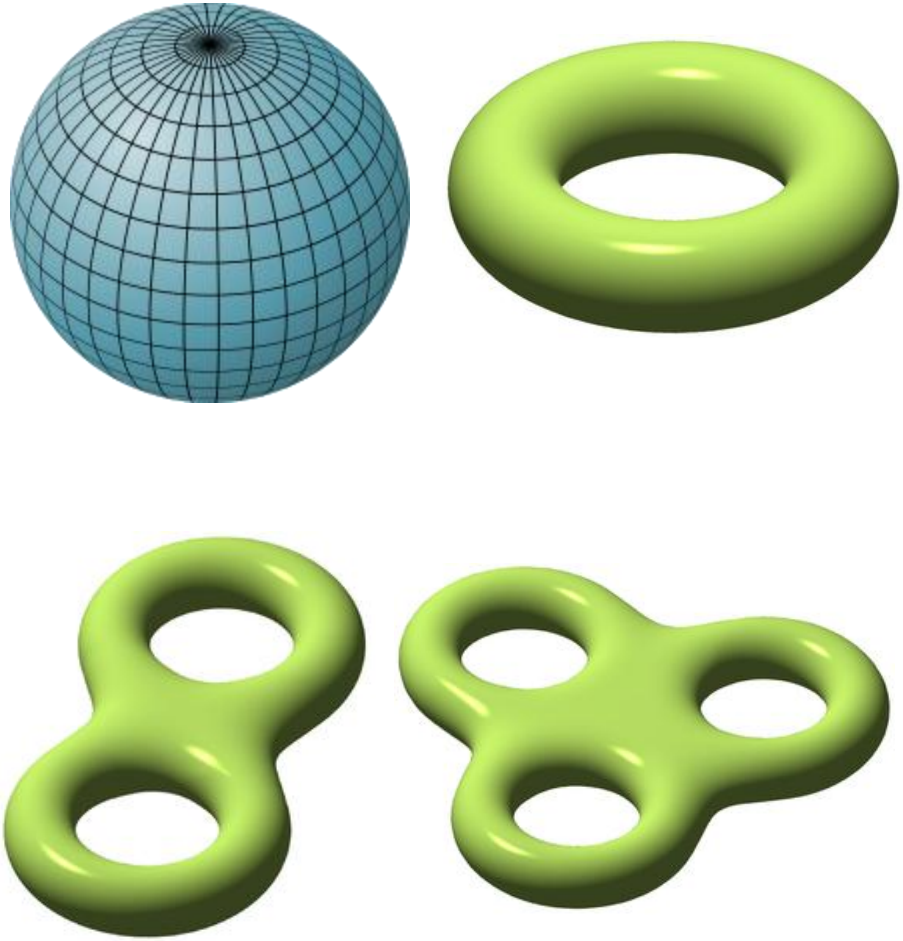
- A finite graph is planar if and only if it does not contain a subgraph that is a ***subdivision*** of K_5 (the complete graph on five vertices) or of $K_{3,3}$ (utility graph).





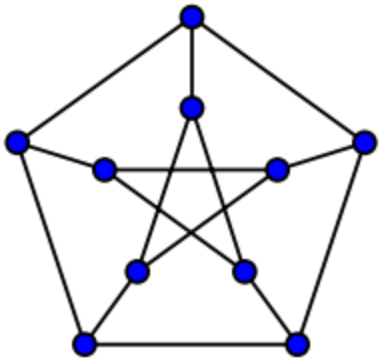
Graph Theory and Topology

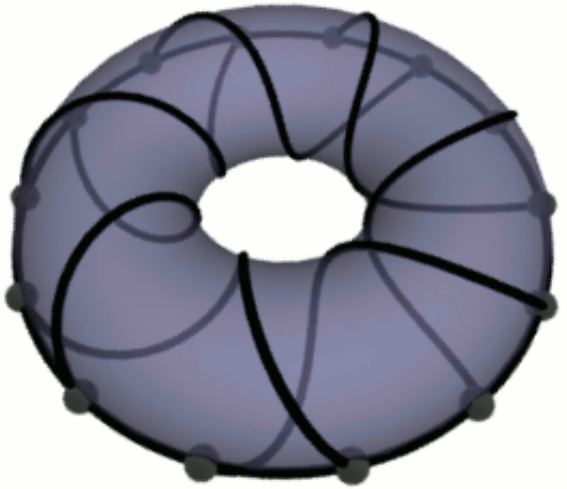
- Planar graphs can be drawn on a sphere.
- Graphs can be embedded on other 2 dimensional surfaces (2-Manifolds)
- Genus of a graph is the genus of the surface it is embeddable on with no crossings [confirm]
- Genus 0, 1, 2, 3

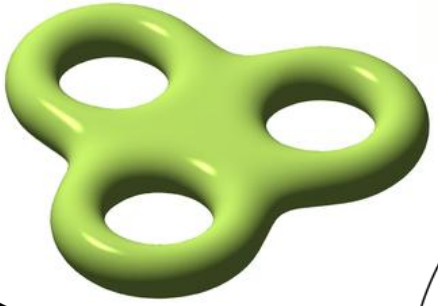


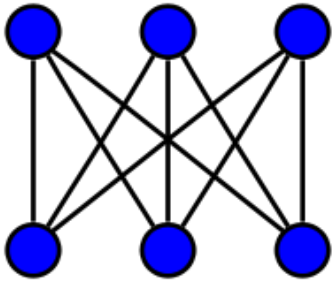
Graph Theory and Topology

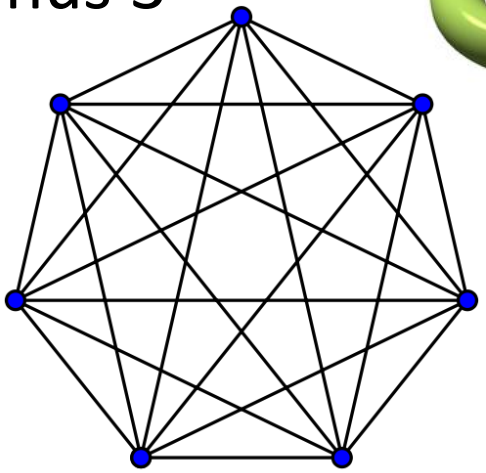
- Toroidal Graphs
- Just as graphs can be planar they can be toroidal
- $K_{3,3}$, K_7, K_6, Petersen
- Klein, Genus 3

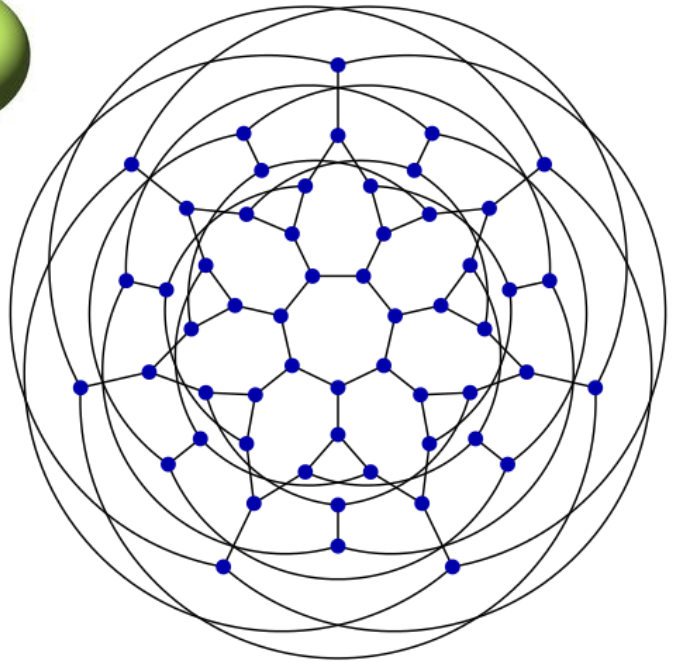






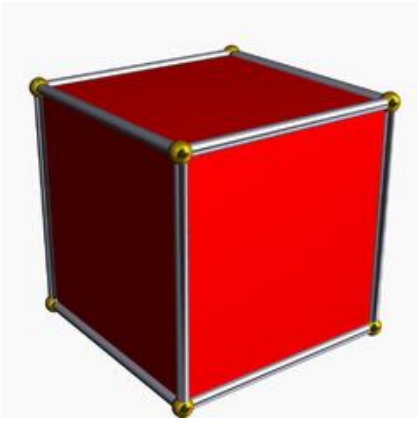


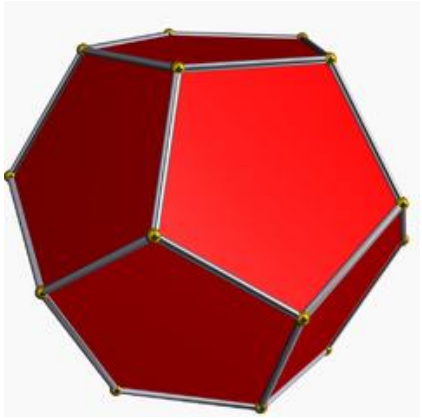


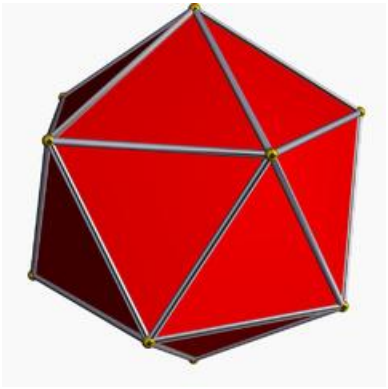


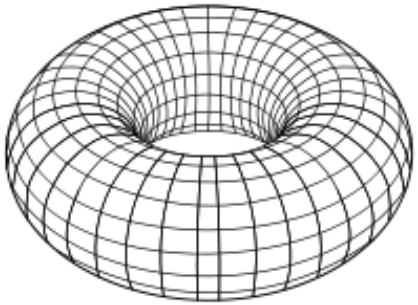
Graph Theory and Topology

- Euler Characteristic
- Convex Polyhedra
- $V - E + F = 2 - 2g$





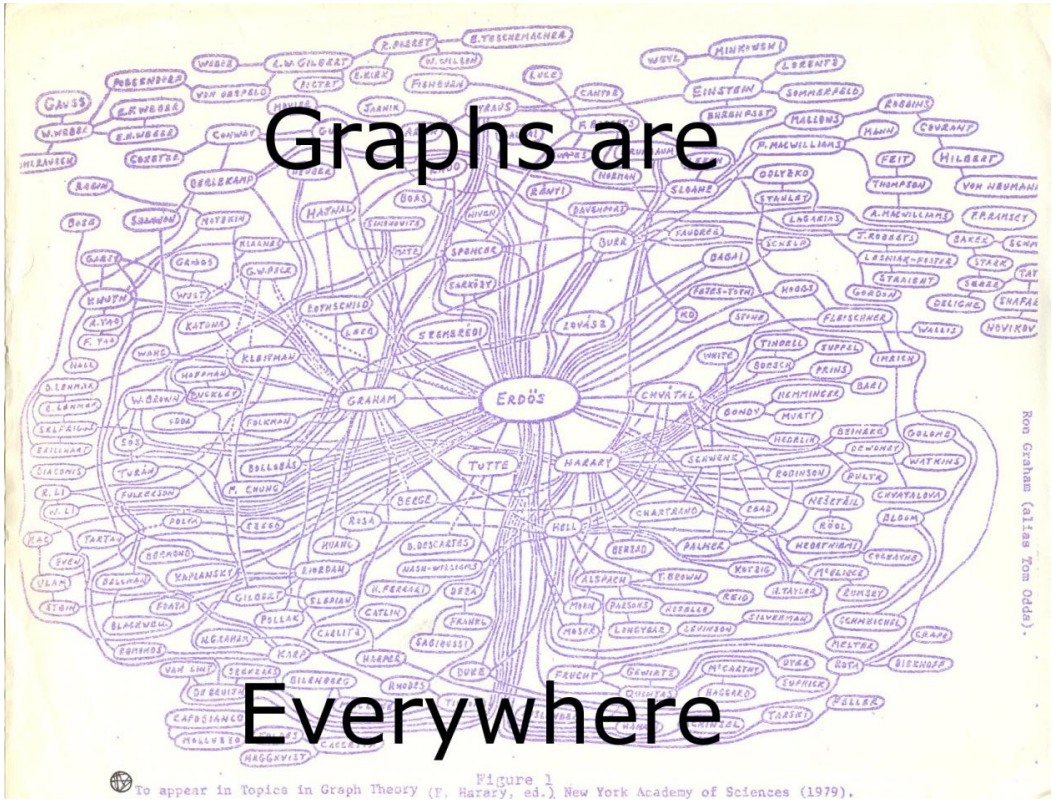




Graphs: Labeling and Data

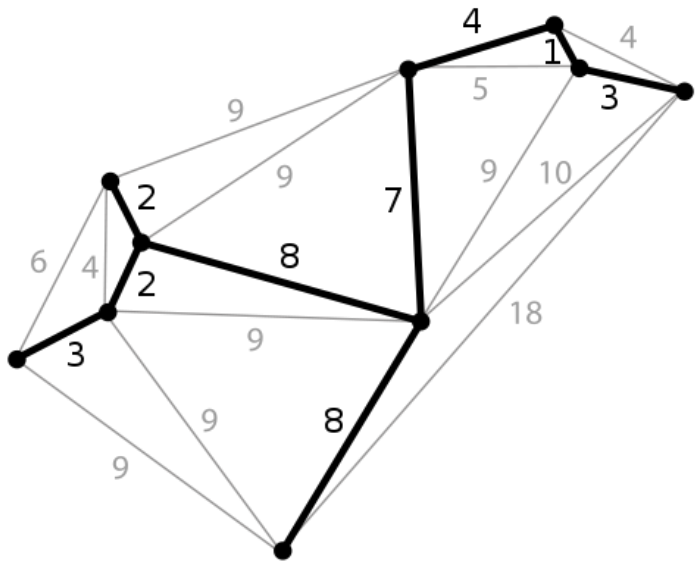
- Graphs are interesting mathematical objects and can be thought solely as such. Graph labeling is a pure mathematical discipline. That is not what we are interested in here.

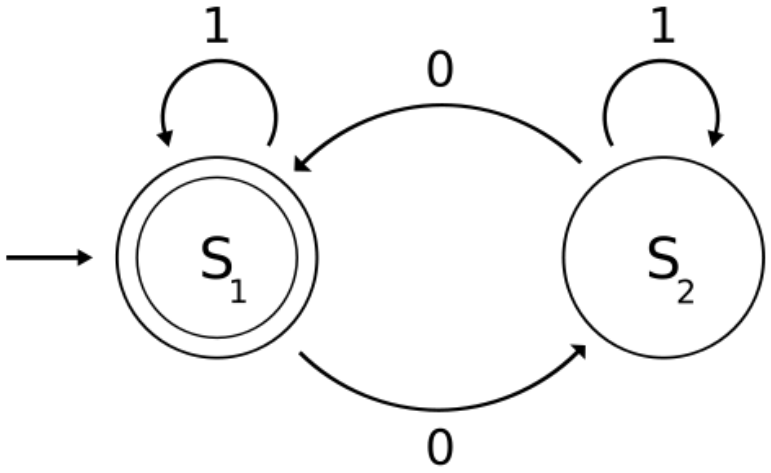
- Property Graph.
- Graphs are increasingly used in data analysis and data mining and are becoming more important if not critical in many business and scientific applications.
- Graphs allow data to be organized, structured and analyzed using graph based approaches, hence the math becomes useful for real world applications.



Graphs: Labeling and Data

- Attaching data to a graph:
- Theoretically any abstract data can be attached to both edges and vertices. [Labeling]
- **Weighted Graphs** are graphs with numerical "weights" attached to edges
- Finite State Automata: vertices are labeled states, edges are labeled with transitions
- Many examples in math and the real world



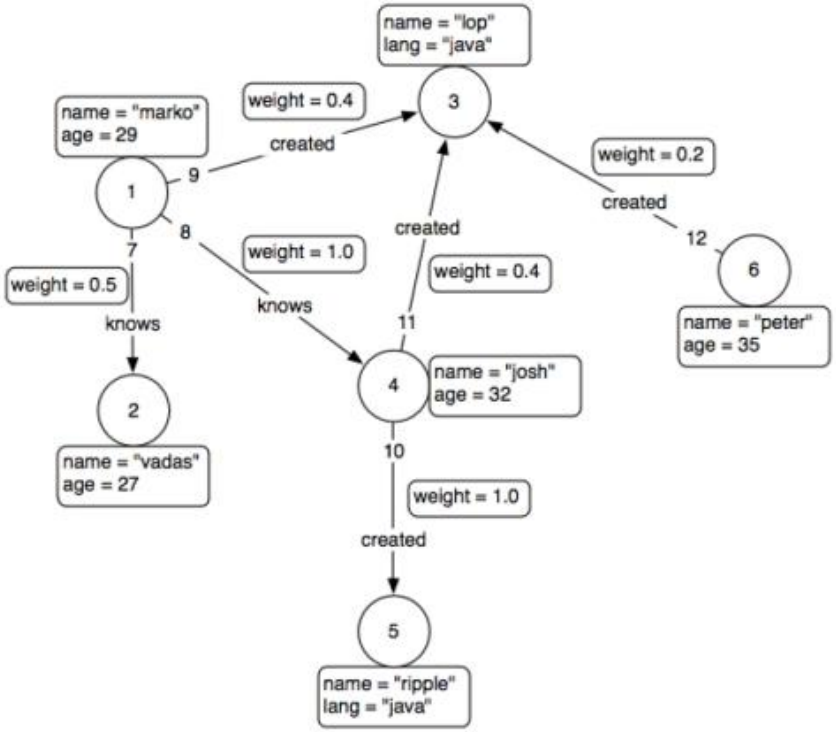


Graphs: Labeling and Data

- The Neo4J Property Graph: Datatypes
- Numbers: Integer values, with capacity as Java's Long type, and floating points, with capacity as Java's Double.
- Booleans.
- Strings.
- Arrays of the basic types above.

Graphs: Labeling and Data

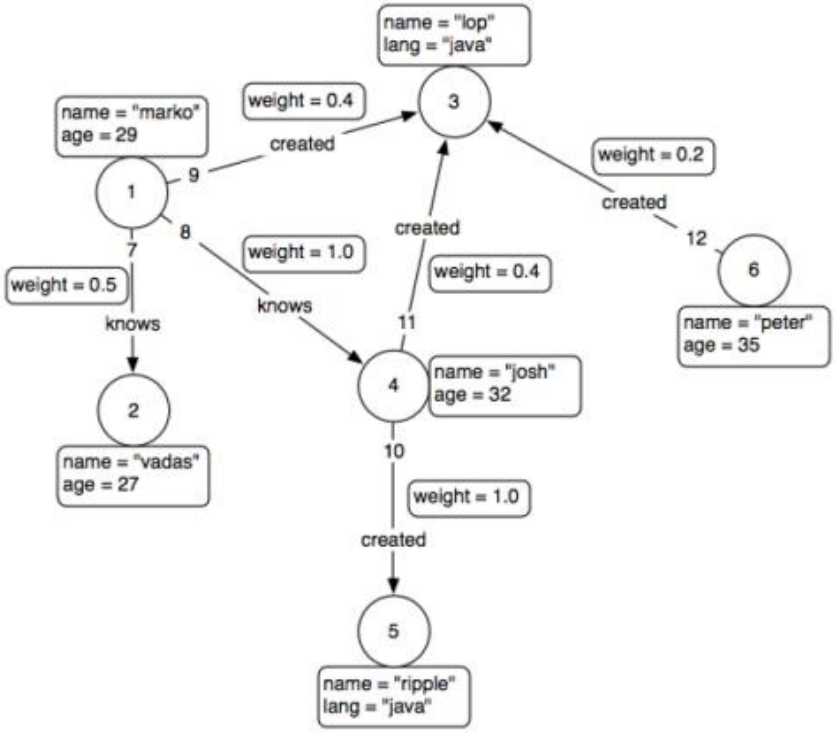
- The Neo4J Property Graph: Nodes
- Node = Vertex
- Multiple **Labels** can be attached to a Node. They are atomic strings (lexical items) which can be queried exactly, no partial matching.
- Properties can be attached to a Node. Properties are name value pairs where the name is a string and the values conform to the allowed Neo4J datatypes.



neotechnology

Graphs: Labeling and Data

- The Neo4J Property Graph: Relationships
- Relationship = Edge
- A single **Type** can be attached to a Relationship. It is an atomic string (lexical item). Can be queried exactly, no partial matching.
- Properties can be attached to a Relationship. Properties are name value pairs where the name is a string and the values conform to the allowed Neo4J datatypes.



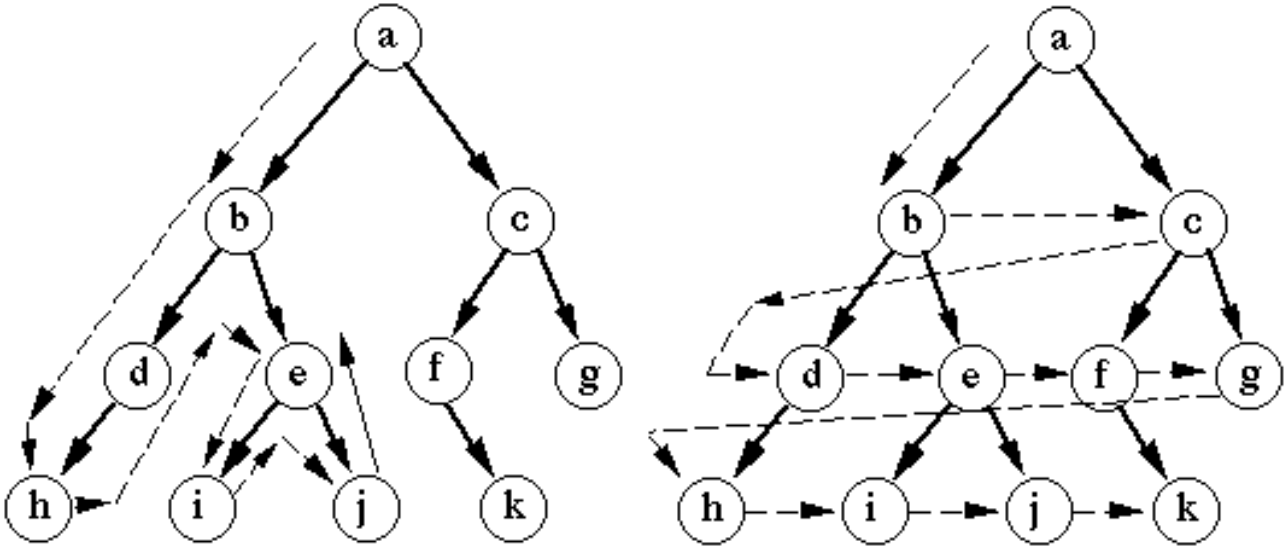
neotechnology

Graphs: Labeling and Data

- Graph Theoretic thoughts about the Neo4J Property Graph: Using graph theory to think about data modeling concerns.
- Relationships are directional, directed edges. This **couples** data modeling with how data queries are constructed.
- Nodes will most likely have a multi-relational (multigraph) structure. There are multiple ways to deal with this. Multiple properties can be attached to a single relationship. How these are modeled can affect query efficiency.
- Data may fall into subgraphs or subdomains which can be “tagged” with Labels and Types

Graph Traversals

- Graph traversal is the problem of visiting all the nodes in a graph in a particular manner.
- Algorithms



Depth-first search

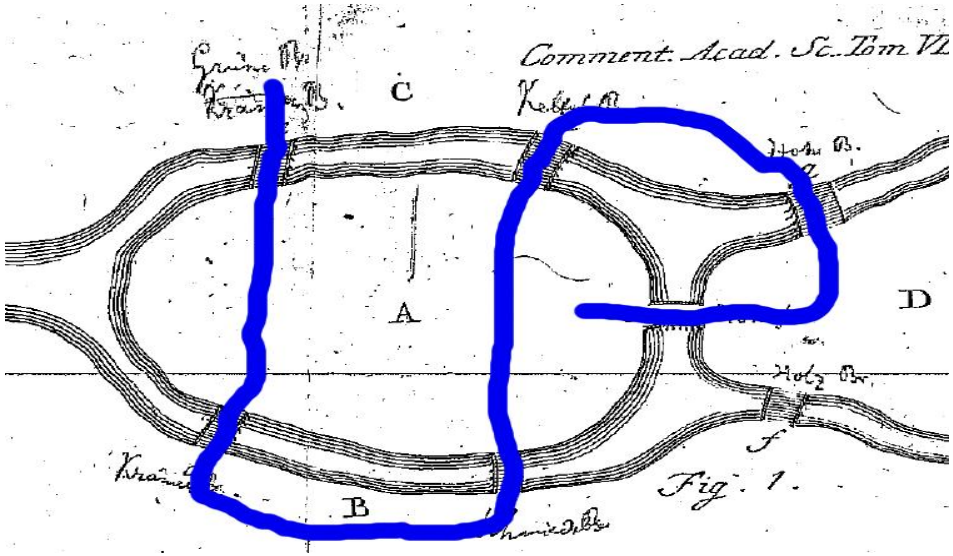
Breadth-first search

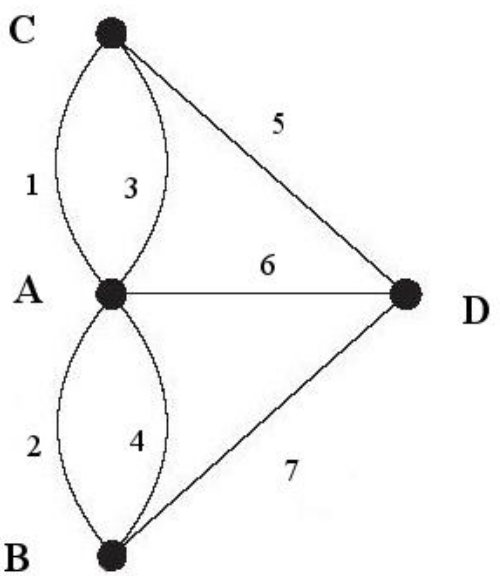
Graph Perambulations

- Types of “Travels” on Connected Graphs
- Warning: Nomenclature may vary
- Walks, Trails, Paths, Circuits, Cycles
- Traveling on the metro

Graph Perambulations : Walks

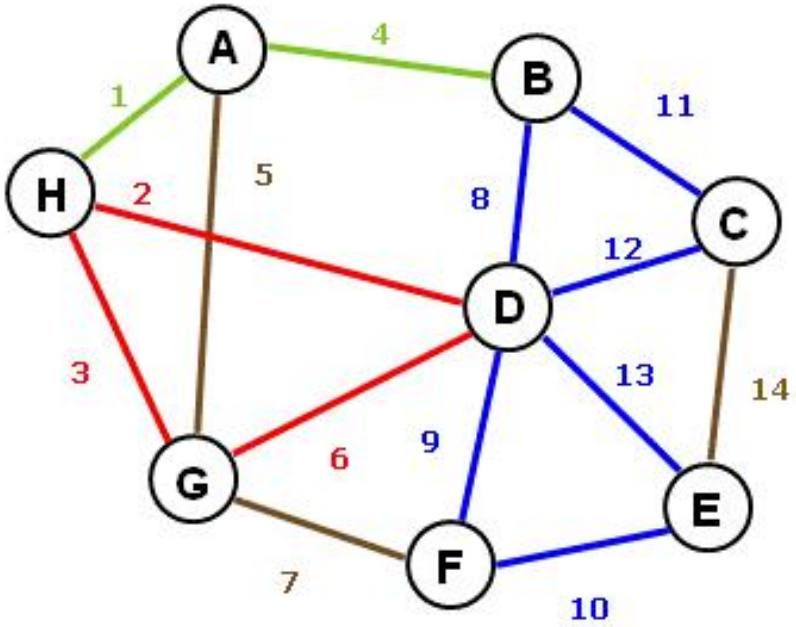
- A walk is a sequence of vertices and edges which has a length, n : (v_0, e_1, v_1, e_2, . . . , v_{n-1}, en, v_n). It is closed if v_0 = v_n. Edges and vertices can repeat. For each j, the edge e_j is incident to v_{j-1} to v_j . If u = v_0 and w = v_n then we call e a uv-walk or a walk from u to v
- ca-walk (**C**,1,**A**,2,**B**,4,**A**,3,**C**,5,**D**,6,B)



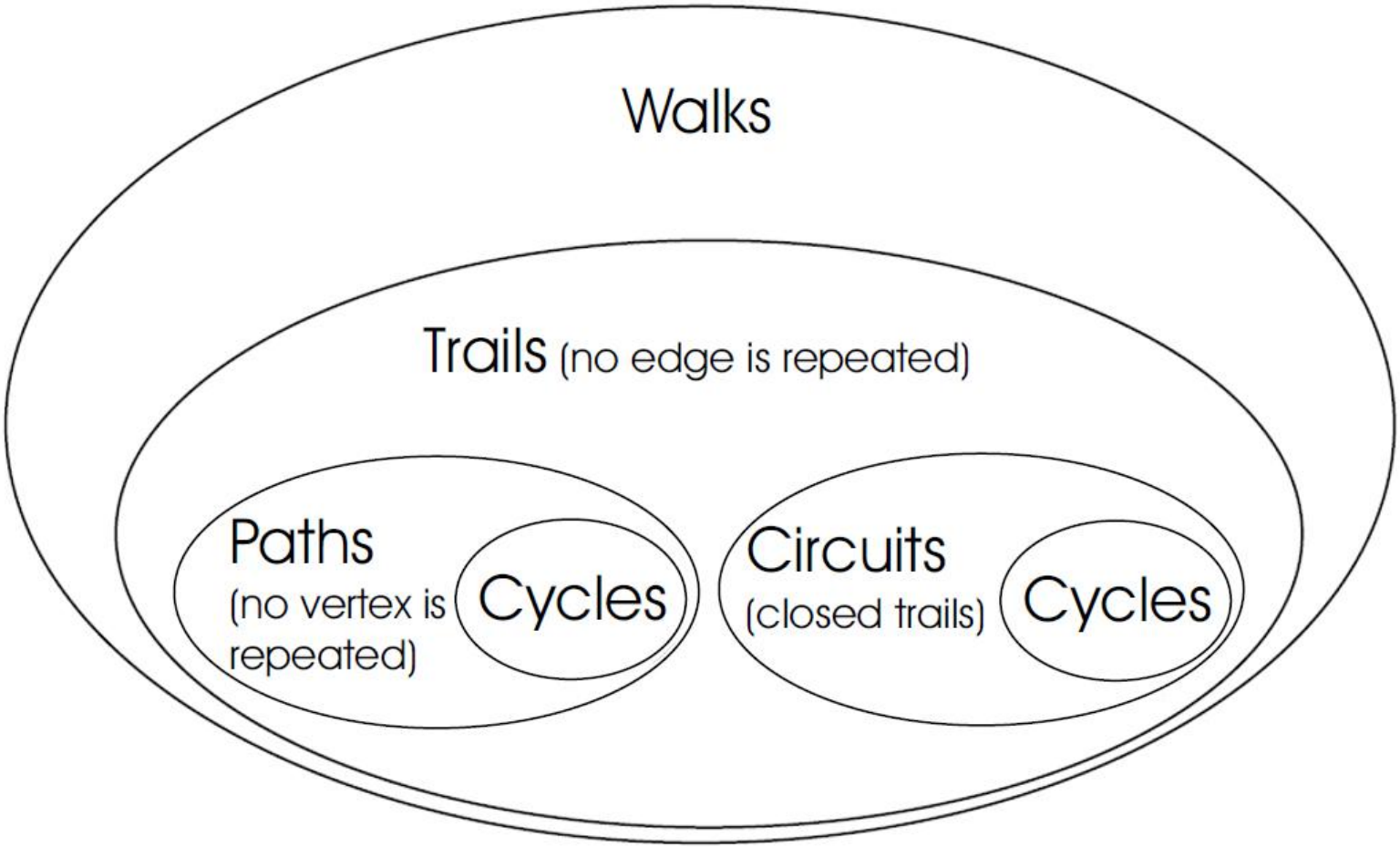


Graph Perambulations

- Trails : A trail is a walk that does not repeat any edge. **H**,2,**D**,13,**E**,10,**F**,9,**D**,12,**C**
- Paths : A path is a walk that does not repeat an vertex. (Paths are Trails) **B**,8,**D**,9,**F**
- Circuits : A closed trail.
- **H**,3,**G**,6,**D**,13,**E**,14,**C**,12,**D**,2,**H**
- Cycles : A closed path. (Cycles are circuits) **B**,8,**D**,12,**C**,11,**B**

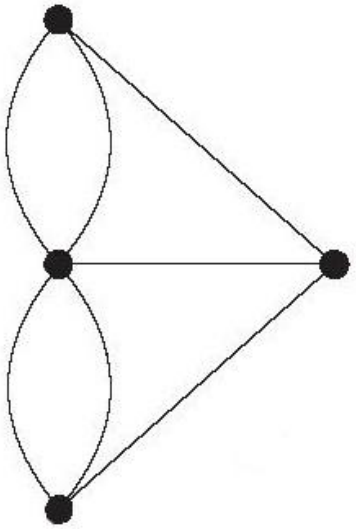


Graph Perambulations

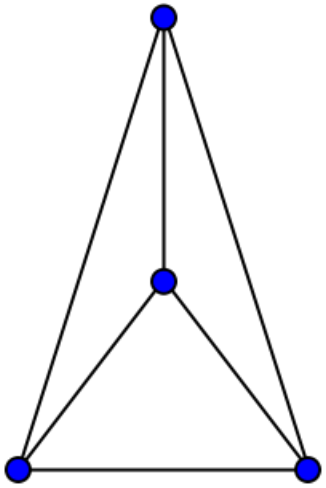


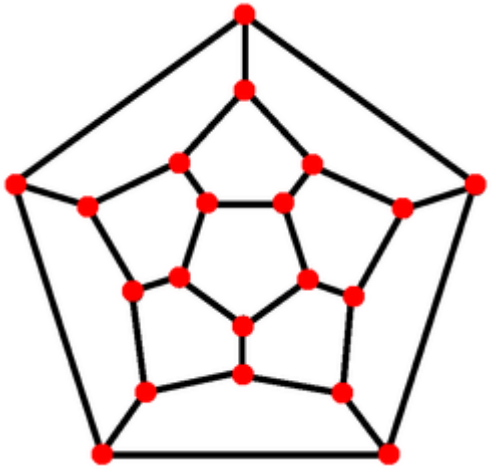
Graph Perambulations

- **Eulerian Trail** - is a trail in a graph which visits every edge exactly once - every vertex has even degree, **Eulerian Circuit** if and only if every vertex has equal in degree and out degree



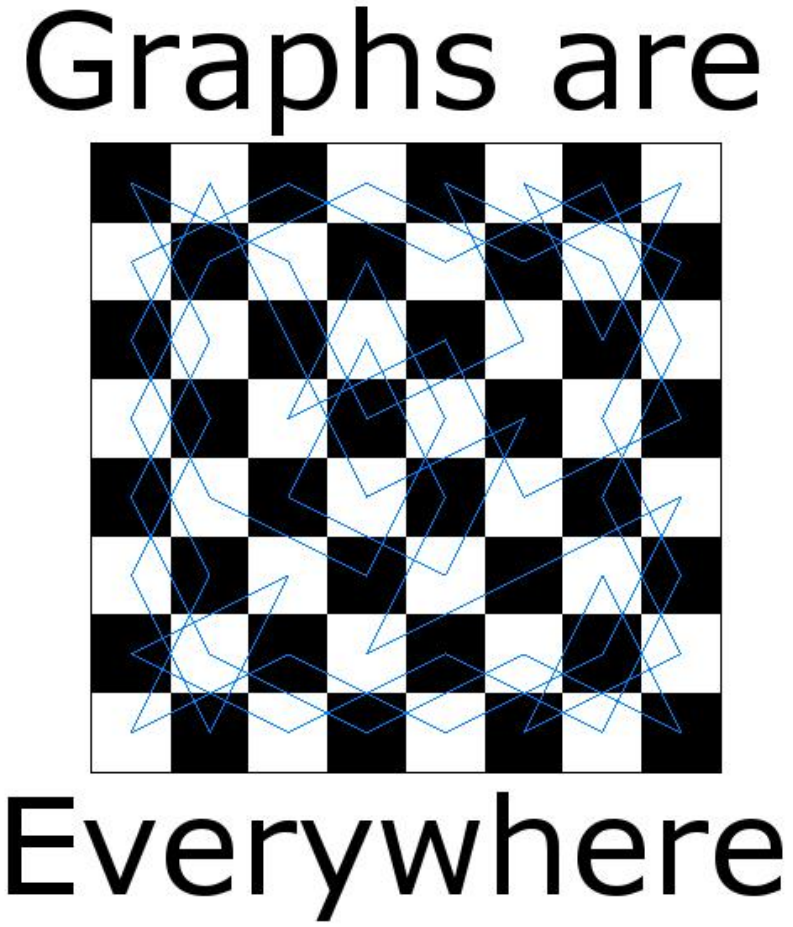
- **Hamiltonian Path** (or traceable path) is a path in an undirected or directed graph that visits each vertex exactly once. A **Hamiltonian Cycle** (or Hamiltonian circuit) is a Hamiltonian path that is a cycle. **Hamiltonian Cycle** is a special case of TSP





Subgraphs and Spans

- Subgraphs are an important area in the study of graph theory and have many important applications in real world analysis



Subgraphs and Spans

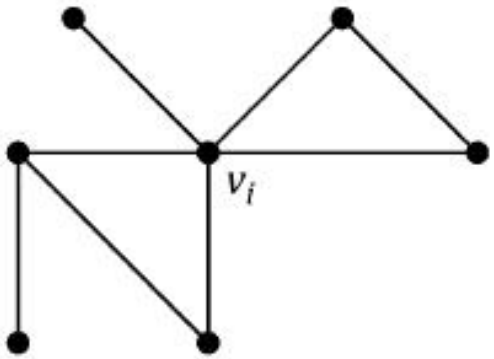
- Since a graph ordered pair G = (V, E) where V is a set V of vertices and E is a set of edges, a subgraph is a subset of those:

$$G' = (V', E') \; where \; V' \subseteq V \; and \; E' \subseteq E$$

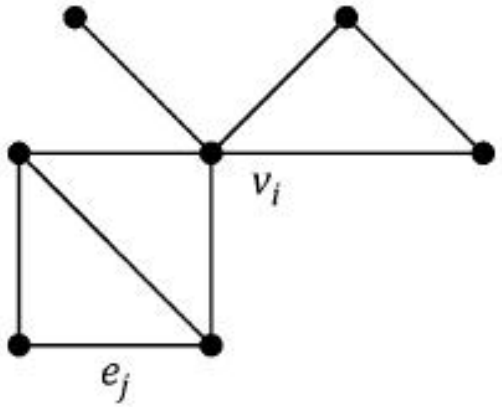
- An easier way to think about a subgraph is in terms of edge and vertex deletions
- Any removed vertex removes all connected edges, in other words every remaining edge has to have both vertices remaining

Subgraphs and Spans

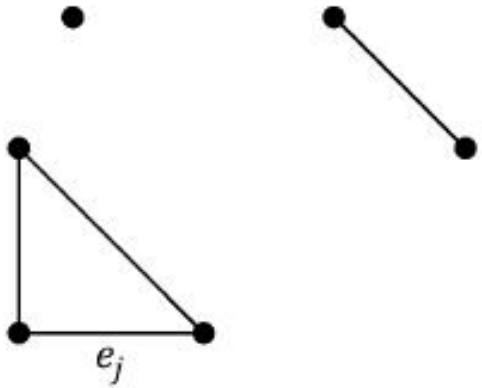
- Creating Subgraphs by deletion



$G - e_j$



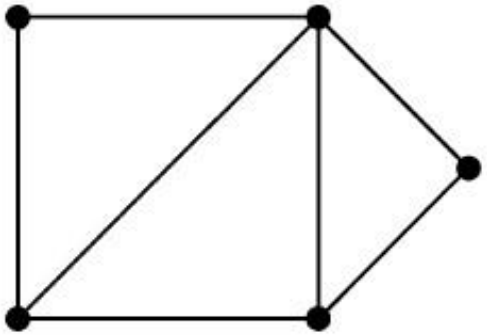
G



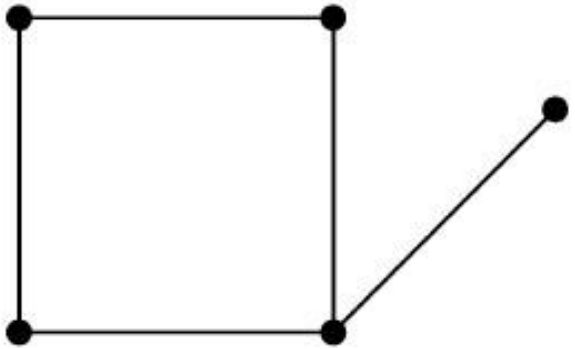
$G - v_i$

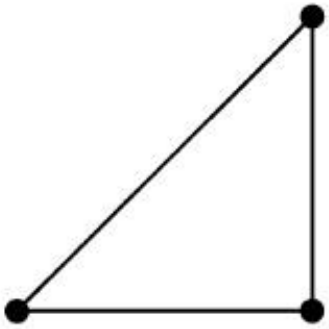
Subgraphs and Spans

- Spanning Subgraph (Includes all vertices from original graph)



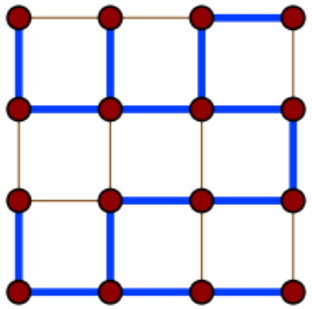
- [Vertex] Induced Subgraph includes a subset of vertices and connected edges

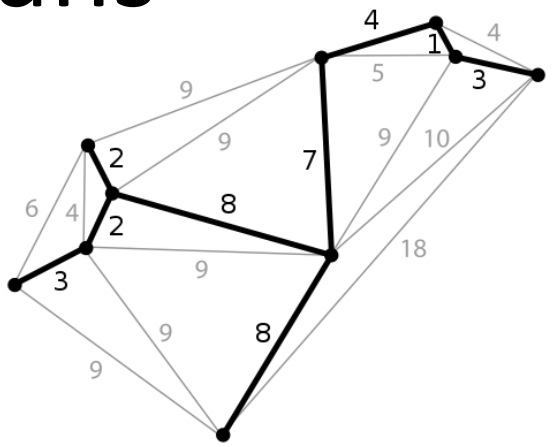




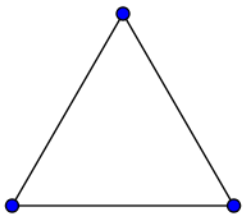
Subgraphs and Spans

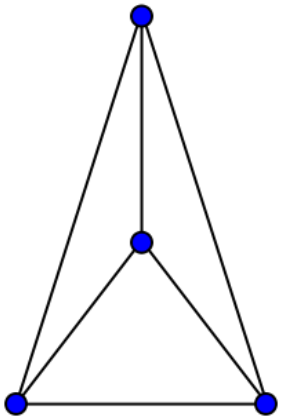
- Spanning Trees
- Trees (acyclic graphs) that include all vertices of a graph
- Minimum Spanning Tree
- Cayley's Theorem
- Kirchhoff's Theorem

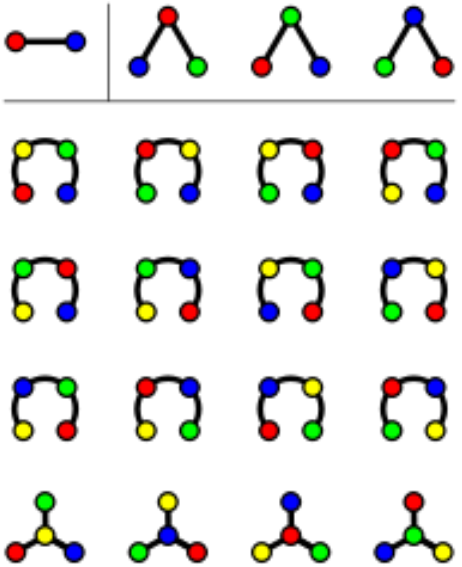




$$n^{n-2}$$



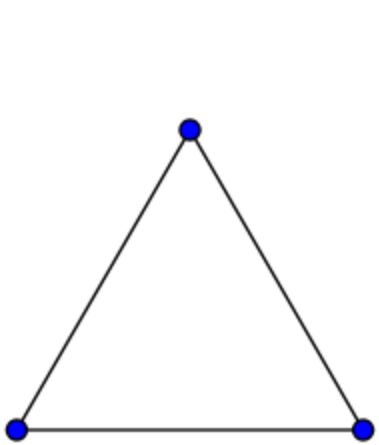


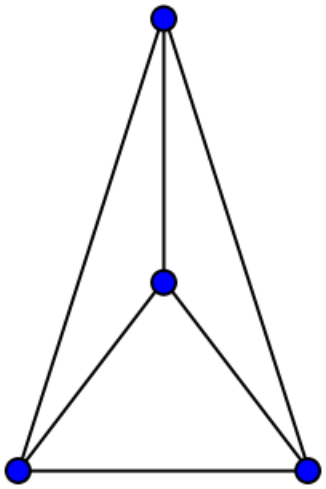


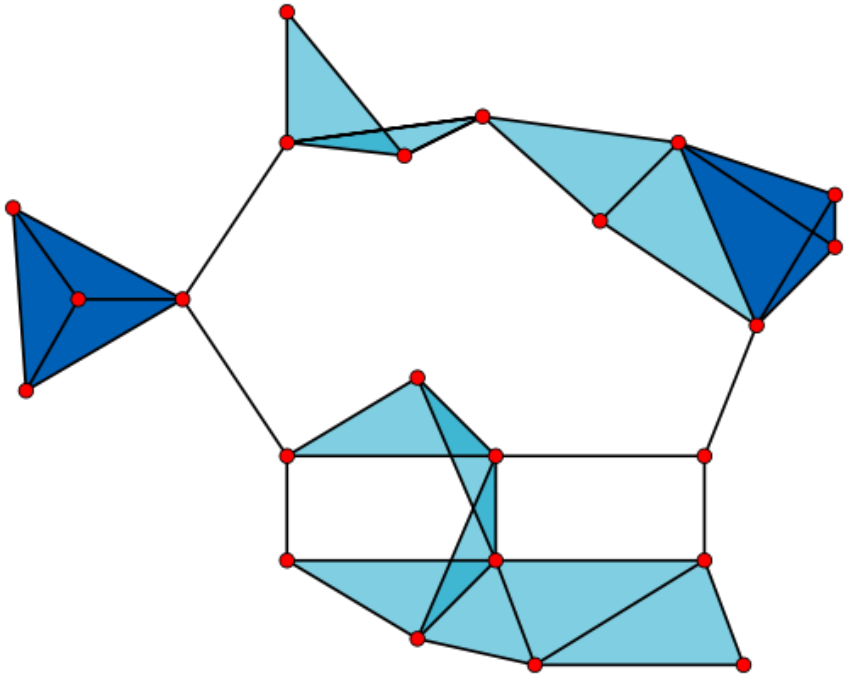
Subgraphs and Spans

- Cliques – Simply put an occurrence of the complete graph as a subgraph
- Complete subgraphs

- K_3, K_4 Cliques





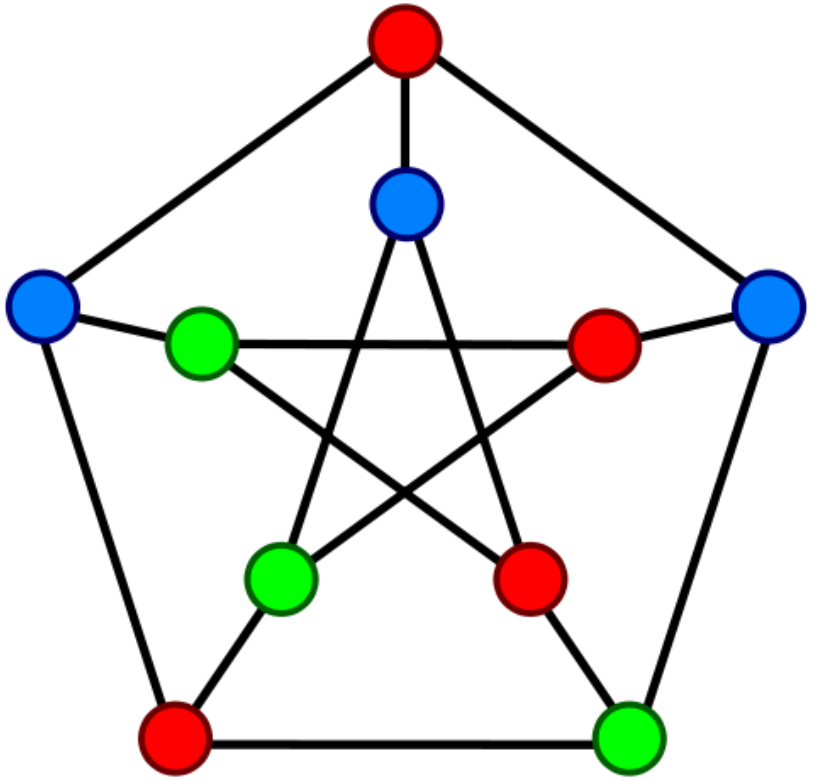


Neo4J and Graph Theory

- Cypher Queries can return Subgraphs
- Neo has a traversal API that can be either Depth first or Breadth first
- Traversal API can return Paths or Subgraphs (Expander Feature)

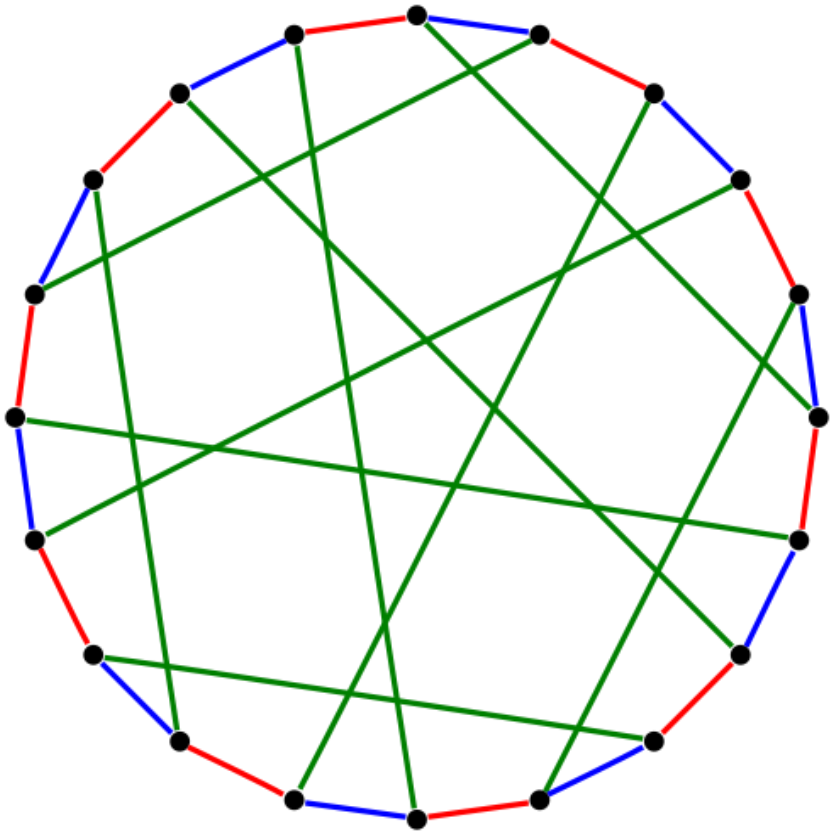
Graph Coloring

- Graph coloring is a special case of graph labeling.
- Same color is not adjacent.
- Vertices and Edges can be colored
- Usually referring to Vertex coloring



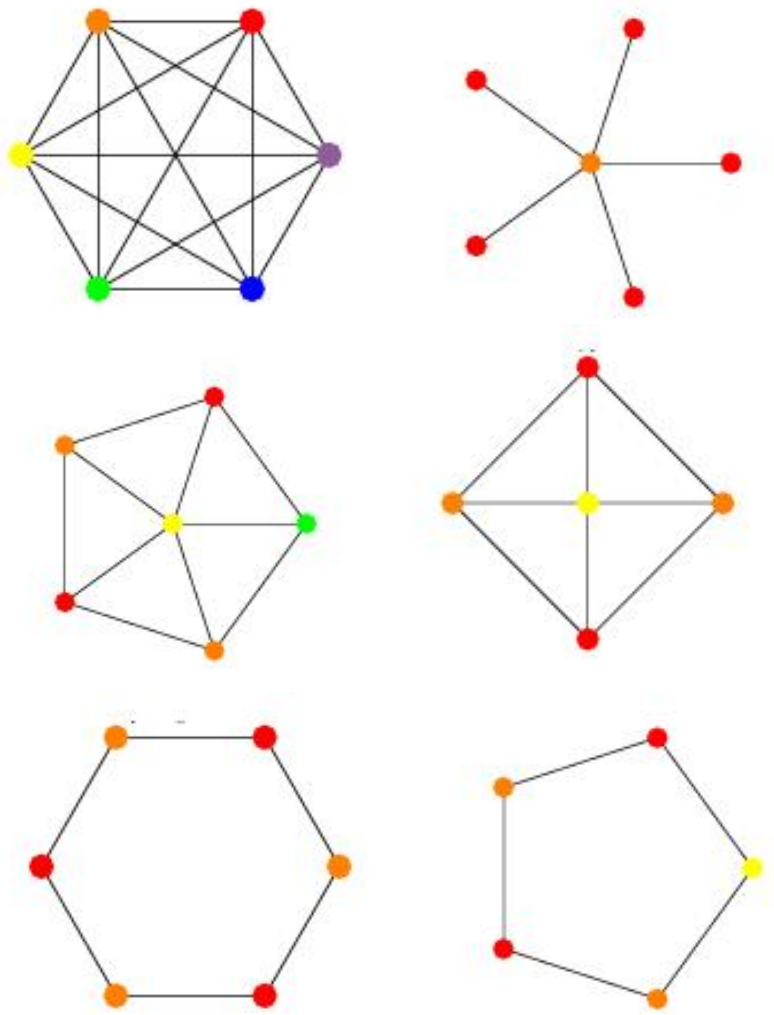
Graph Coloring

- Smallest edge coloring for a graph is called **Chromatic Index**
- By Vizing's theorem, the number of colors needed to edge color a simple graph is either its maximum degree Δ or Δ+1.



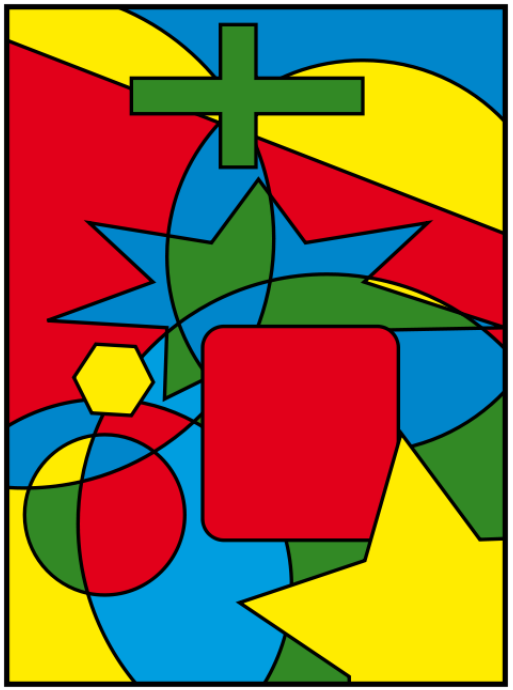
Graph Coloring

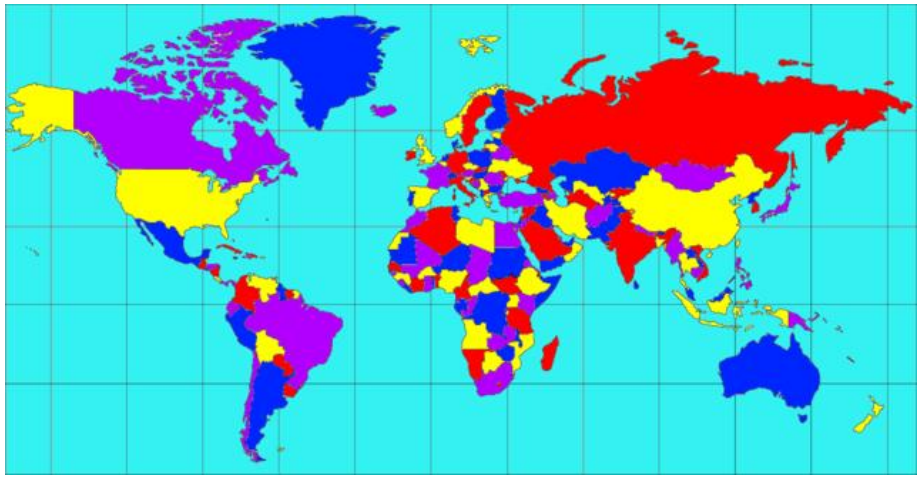
- Smallest vertex coloring for a graph is called **Chromatic Number** denoted $\chi(G)$
- Complete : $\chi(K_n)$ is n
- Star : $\chi(S_n)$ is 2
- Wheel : $\chi(W_n)$ is 4 for even, 3 for odd
- Cycle : $\chi(C_n)$ is 2 for even, 3 for odd

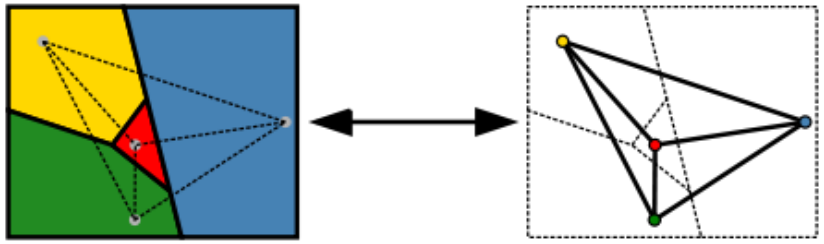


Graph Coloring

- Four Color Theorem
- No more than four colors are required to color the regions of the map (a plane into separated into contiguous regions)
- First major theorem proved by computer

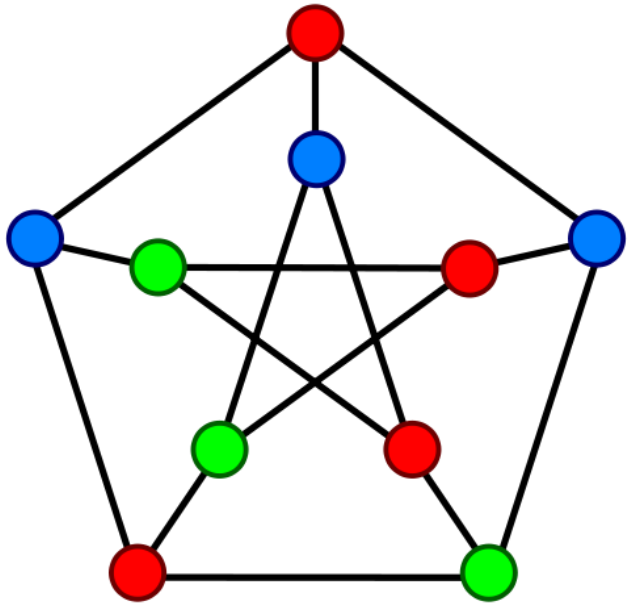




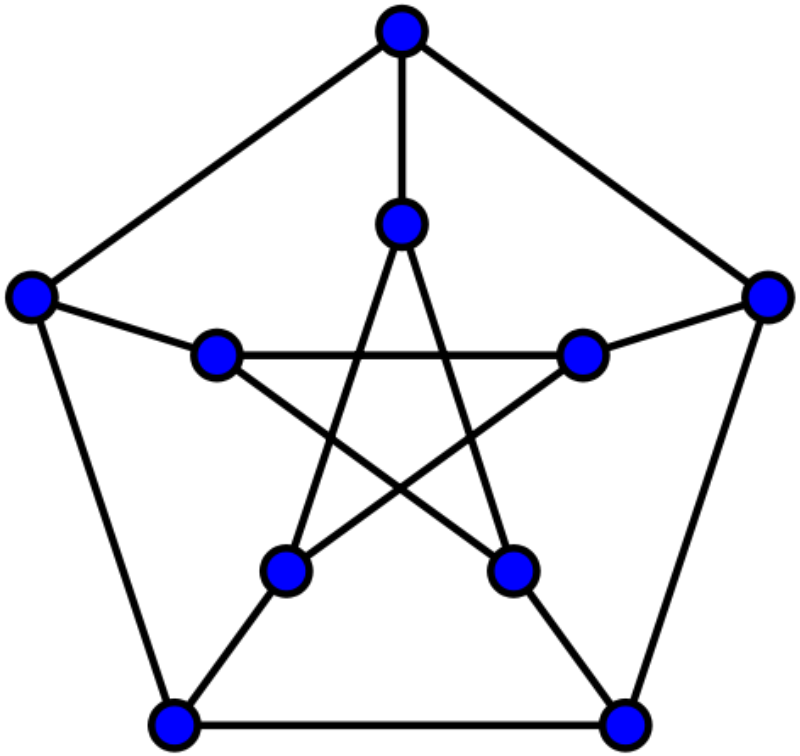


Graph and Vertex Properties

- Graphs and Vertices have qualities usually numeric measures that are interesting and tell us things about them.
- Some graph properties are known as graph invariants and is a property of graphs that depends only on the abstract structure, not on graph representations such as particular labellings or drawings of the graph.
- Some properties (invariants) so far include genus, chromatic number and chromatic index.



Graph Properties



- Order - the number of vertices
- Size - the number of edges
- Girth - the length of the shortest cycle contained in the graph
- Order: 10
- Size: 10
- Girth: 5
- Spectrum: $(-2)^4$, $(1)^5$,$(3)^1$
- Chromatic Polynomial

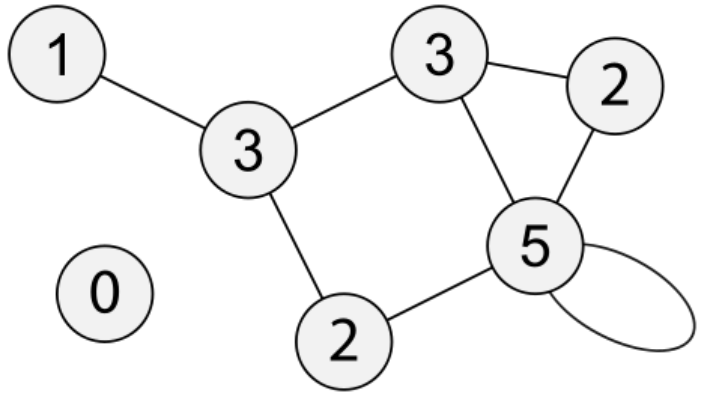
$$t(t-1)(t-2)\left(t^7-12t^6+67t^5-230t^4+529t^3-814t^2+775t-352\right).$$

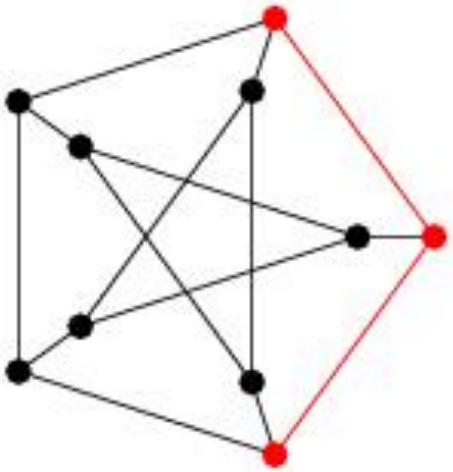
- Characteristic Polynomial

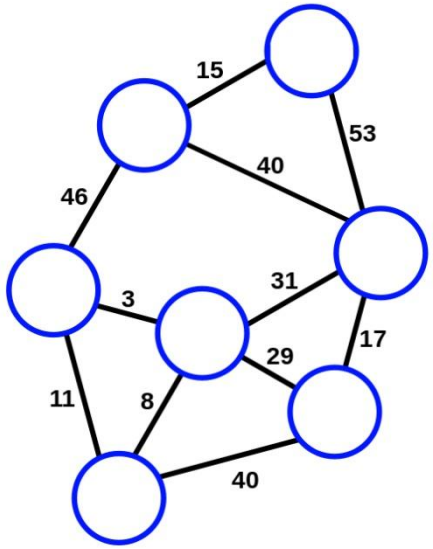
$$(t-1)^5(t+2)^4(t-3)$$

Graph Properties: Vertex Properties

- A Vertex has a degree measure (inbound, outbound)
- The distance between two vertices (graph geodesic distance) is the number of edges in the shortest path between them.
- Distance in weighted graphs is the sum of the weights

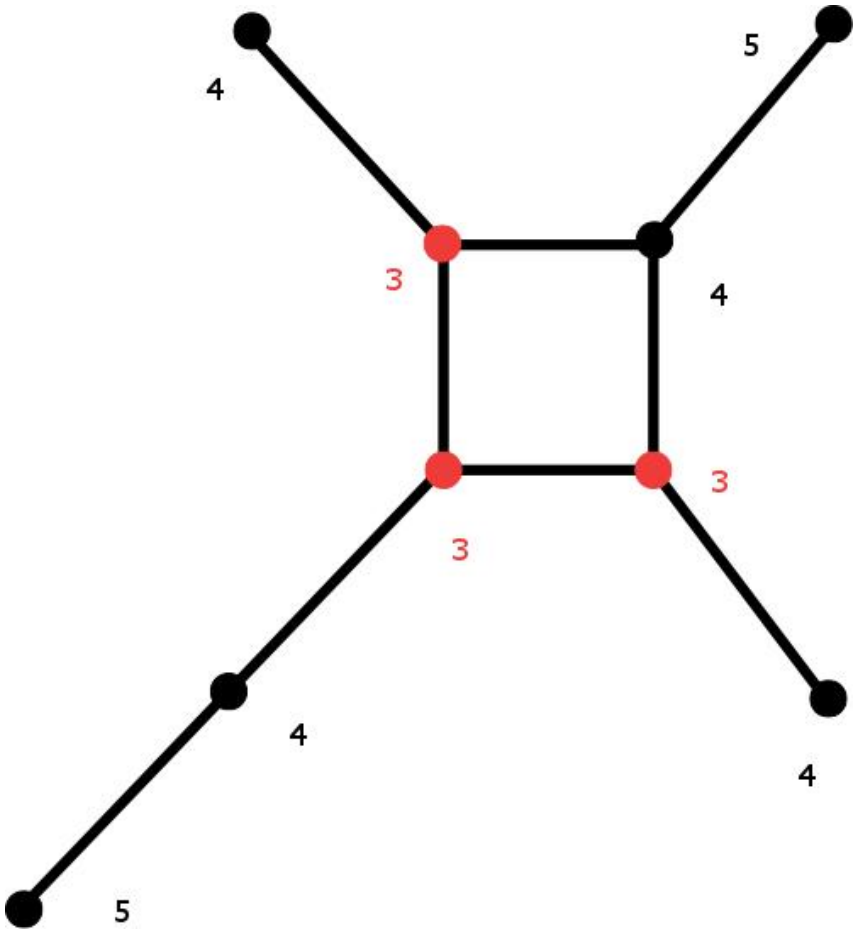






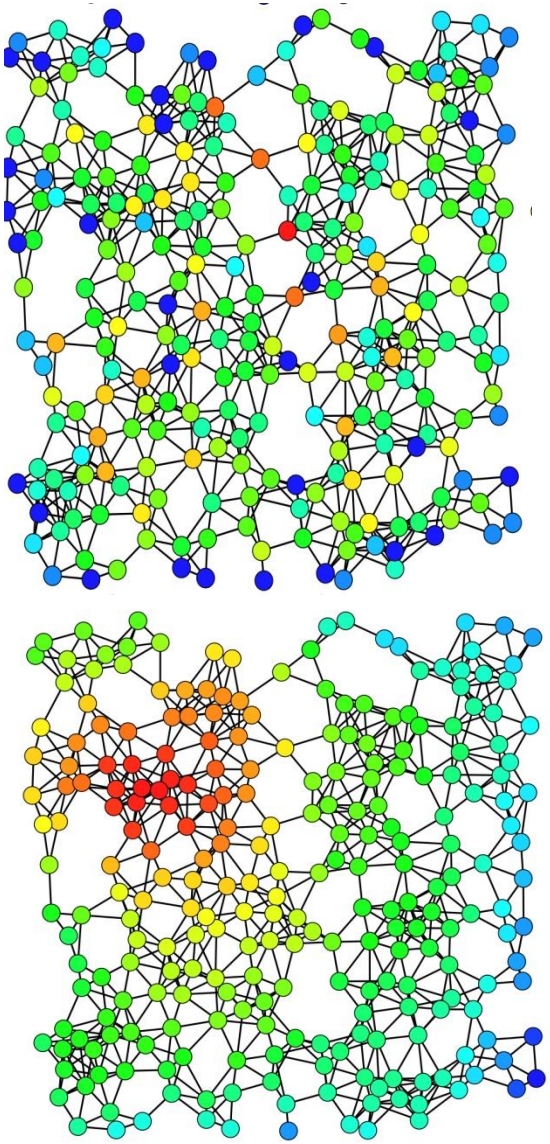
Graph Properties: Vertex Properties

- The **Eccentricity** ε(v) of a vertex v is the greatest geodesic distance between v and any other vertex. It can be thought of as how far a node is from the node most distant from it in the graph.
- The C**enter of a Graph** is the set of all vertices of minimum eccentricity, that is, the set of all vertices A where the greatest distance d(A,B) to other vertices B is minimal



Graph Properties: Vertex Properties

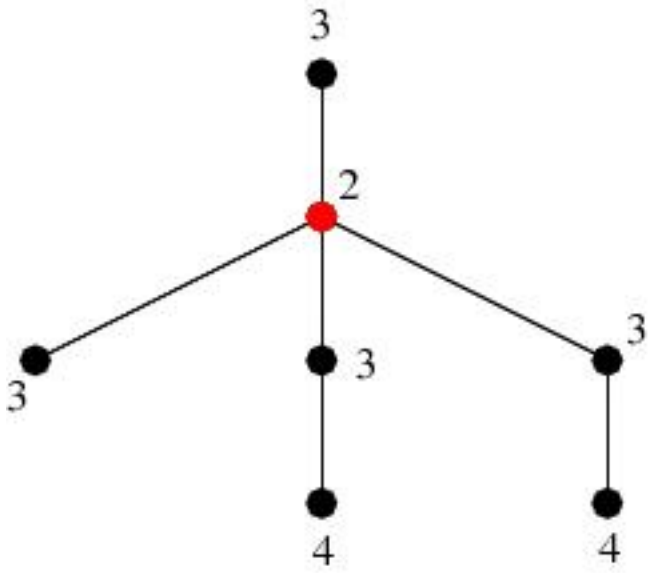
- **Centrality** measures a vertex's relative importance within a graph
- **Betweenness Centrality** quantifies the number of times a node acts as a bridge along the shortest path between two other nodes
- Google's PageRank is a variant of the **Eigenvector Centrality** measure which is a measure of the influence of a node in a network.



Graph Properties

- The **Radius** r of a graph is the minimum eccentricity of any vertex.

- The **Diameter** d of a graph is the maximum eccentricity of any vertex in the graph. That is, d it is the greatest distance between any pair of vertices. To find the diameter of a graph, first find the shortest path between each pair of vertices. The greatest length of any of these paths is the diameter of the graph.



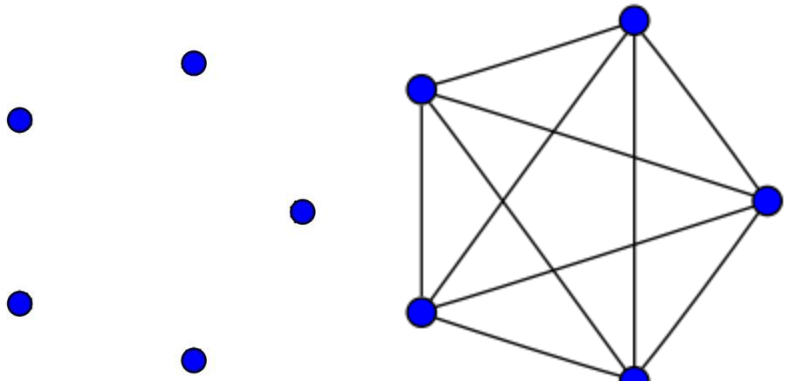
radius = 2
diameter = 4

$$r = \min_{v \in V} \epsilon(v)$$

$$d = \max_{v \in V} \epsilon(v)$$

Graph Properties

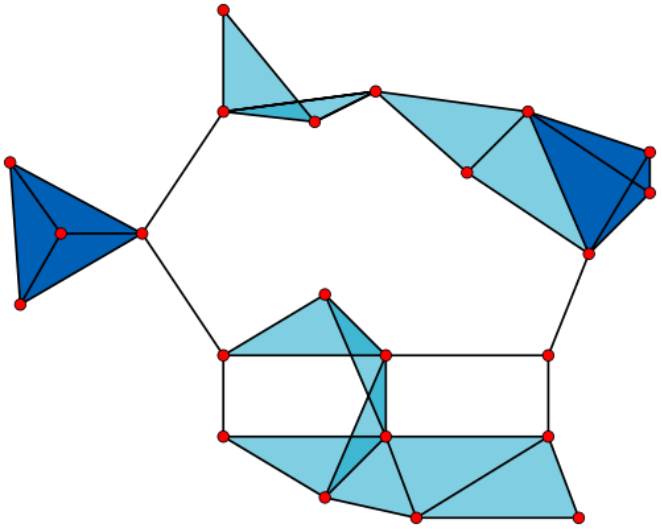
- The **Density** of G is the ratio of edges in G to the maximum possible



- Measure on **Undirected Simple** graphs

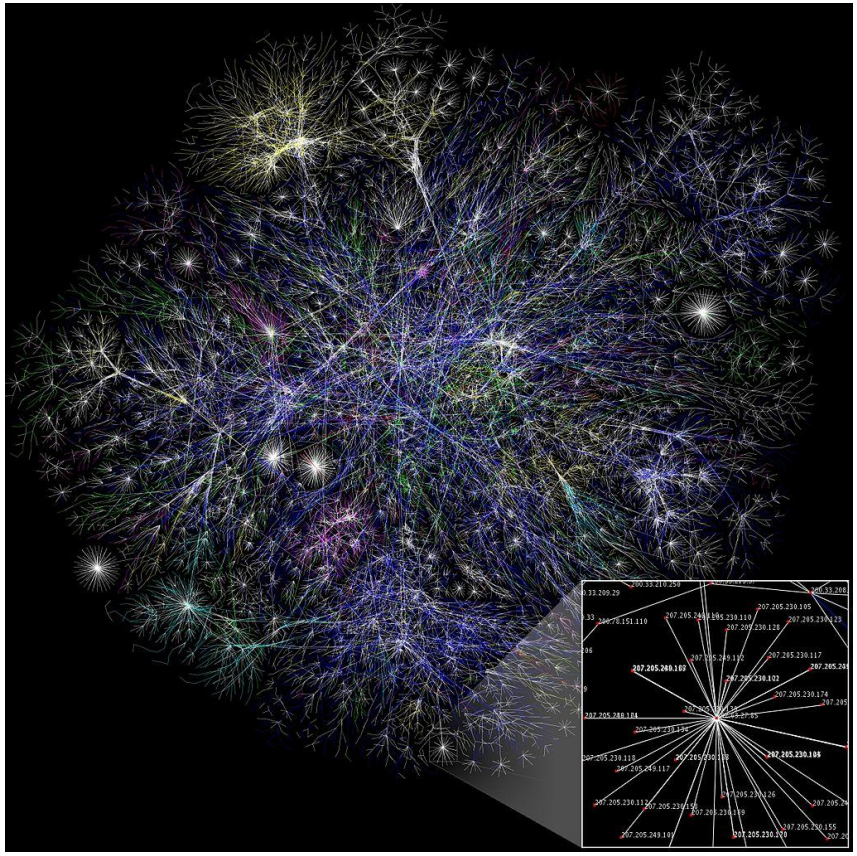
$$D = \frac{2|E|}{|V|\left(|V|-1\right)} \qquad T_{n-1} = \frac{n^2-n}{2}$$

- **Clique number** ω(*G*) of a graph *G*, the order of a largest clique in *G*



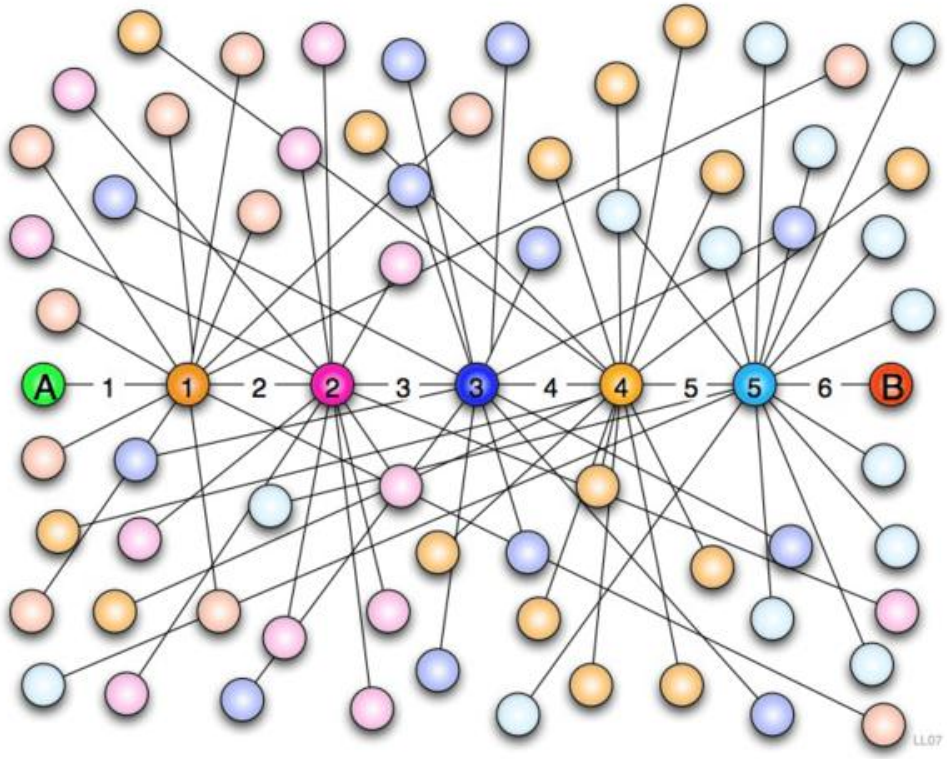
Network Analysis

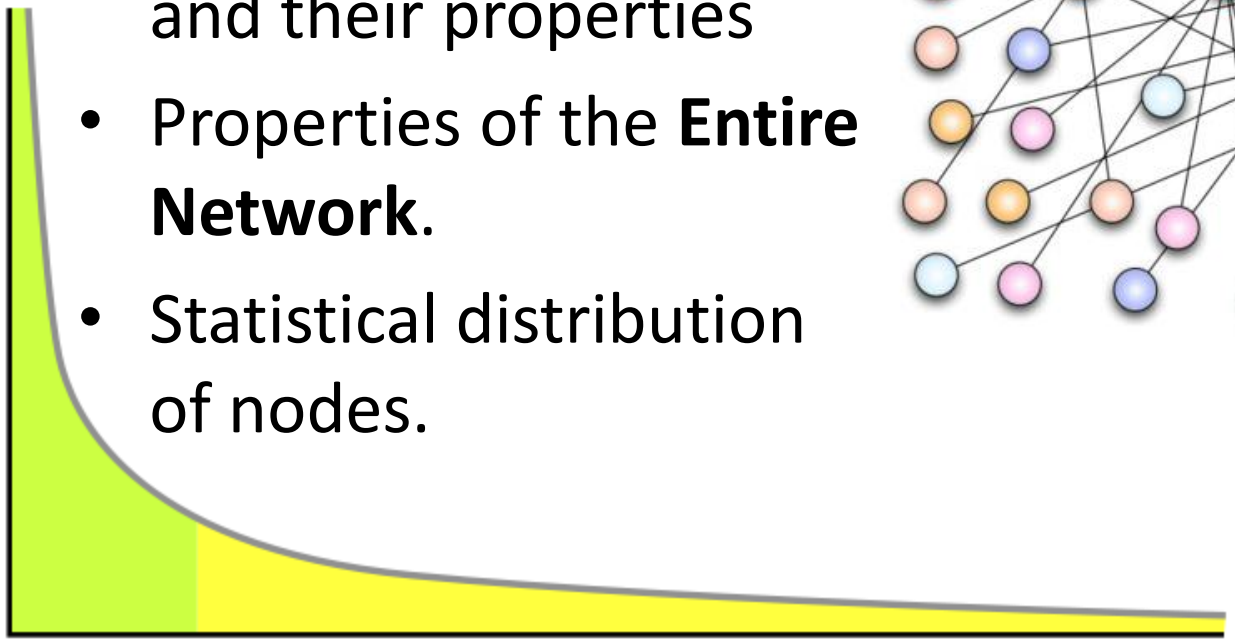
- **Network Analysis** is a sub-discipline of **Graph Theory**
- Replace the word **graph** with **network**
- It has application in many disciplines including statistical physics, particle physics, computer science, electrical engineering, biology, economics, operations research, and sociology.



Network Analysis

- Properties about I**ndividual Nodes**. Link analysis. Centrality
- Identification of **Groups** and their properties
- Properties of the **Entire Network**.
- Statistical distribution of nodes.

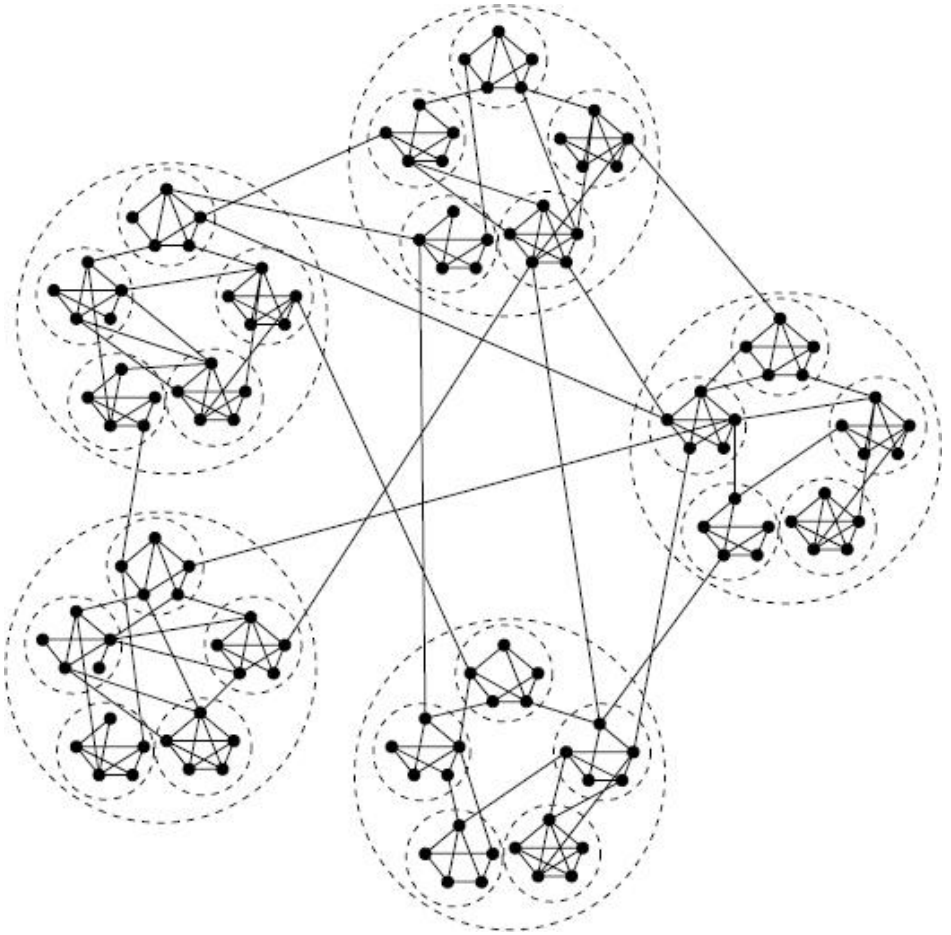




$$y \sim r^{-b}$$

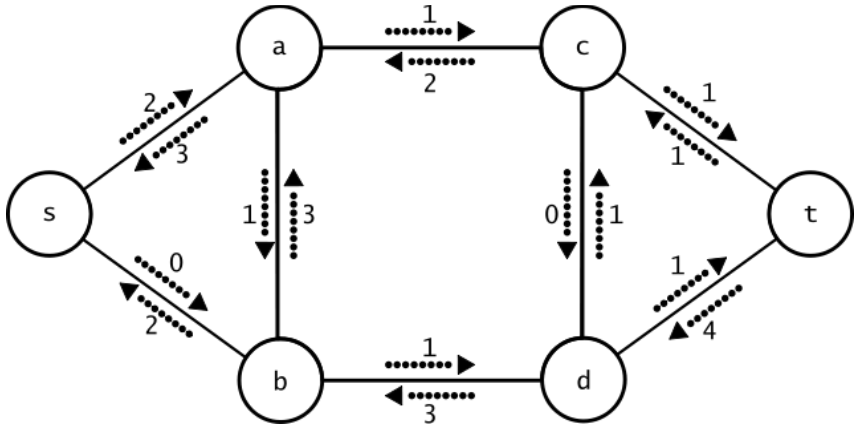
Network Analysis: Clustering

- Dense components (Dense Subgraphs)
- Clusters can be Hierarchical
- Cliques
- Intra-cluster density versus inter-cluster sparsity



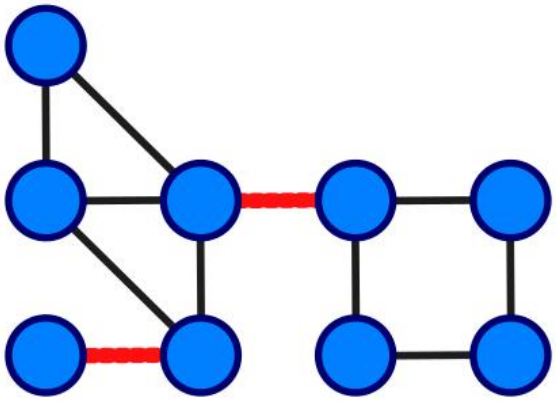
Network Analysis: Flow

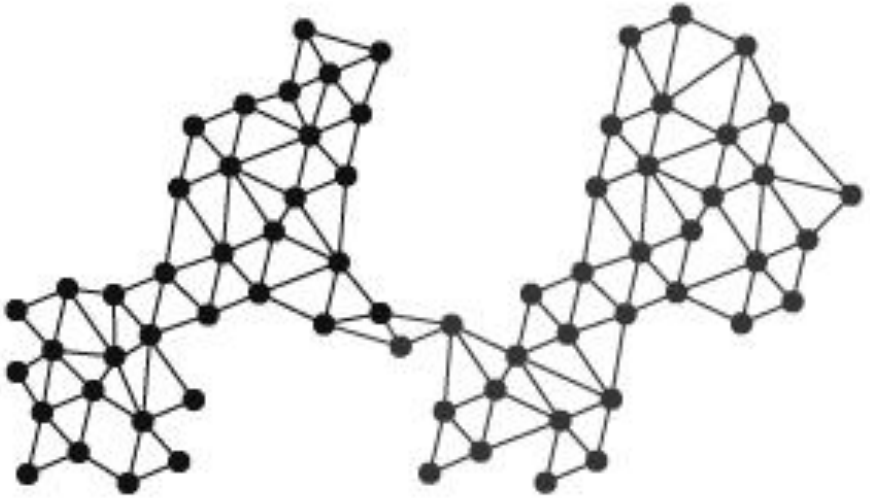
- Analysis of network capacity or routing
- Electricity, fluids, traffic, Operations research, ecological systems
- Matching problems, assignment, stable marriage



Network Analysis: Robustness

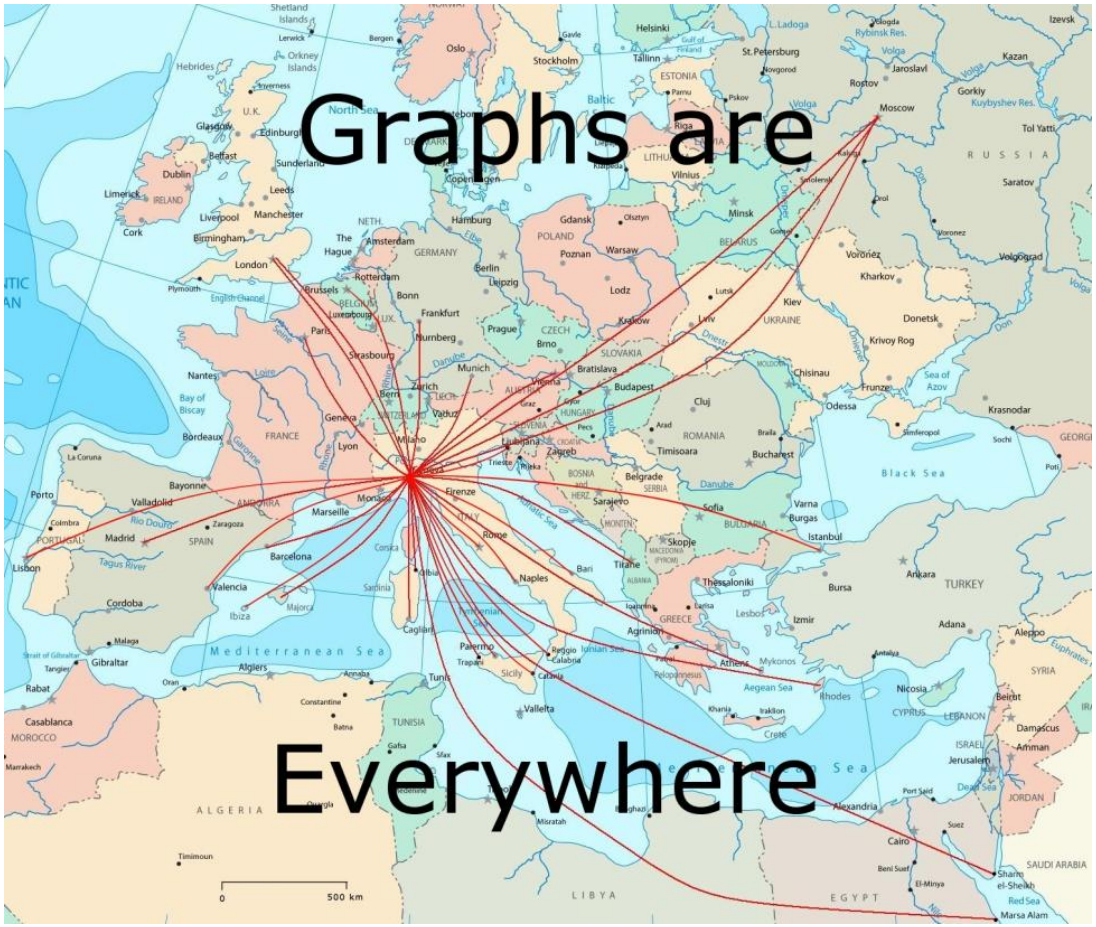
- Edge Cut: Bridge
- Vertex Cut: Cut-point
- Bottlenecks
- Cheeger Constant or
  Isoperimetric Number





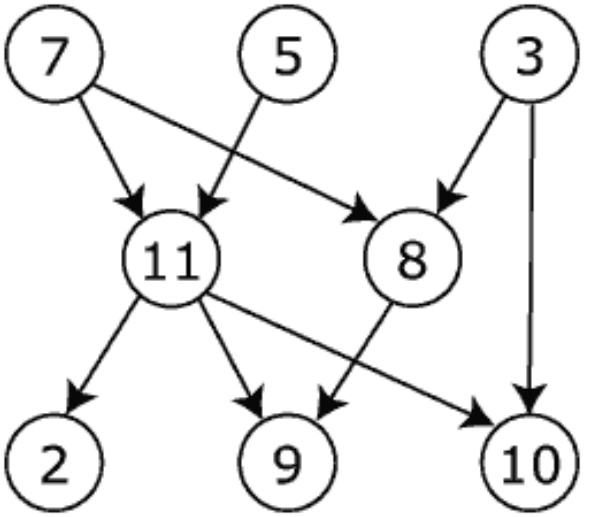
Graph Algorithms

- Algorithms that can be applied to graphs
- Determining properties of graphs
- Finding Elements, Subcomponents (Subgraphs), Walks
- Traversals
- Linear Algebra
- Quite a few NP hard problems (TSP)



Graph Algorithms: Topological Sort

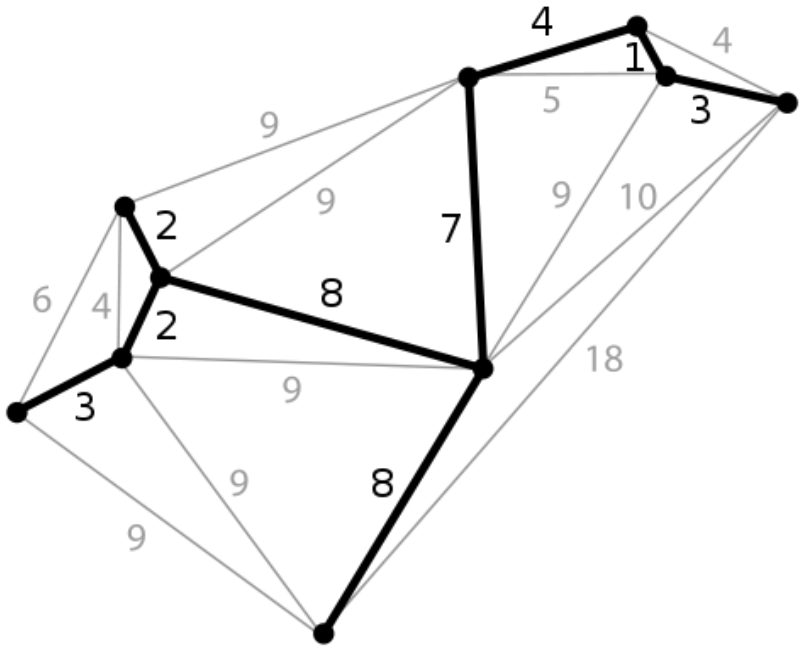
- List vertices such that dependencies are listed subsequently
- Can be used in scheduling a sequence of jobs or tasks based on their dependencies
- O(|V| + |E|)



- 7, 5, 3, 11, 8, 2, 9, 10 (visual left-to-right, top-to-bottom)
- 3, 5, 7, 8, 11, 2, 9, 10 (smallest-numbered available vertex first)
- 5, 7, 3, 8, 11, 10, 9, 2 (fewest edges first)
- 7, 5, 11, 3, 10, 8, 9, 2 (largest-numbered available vertex first)
- 7, 5, 11, 2, 3, 8, 9, 10 (attempting top-to-bottom, left-to-right)
- 3, 7, 8, 5, 11, 10, 2, 9 (arbitrary)

Graph Algorithms: Minimum spanning trees

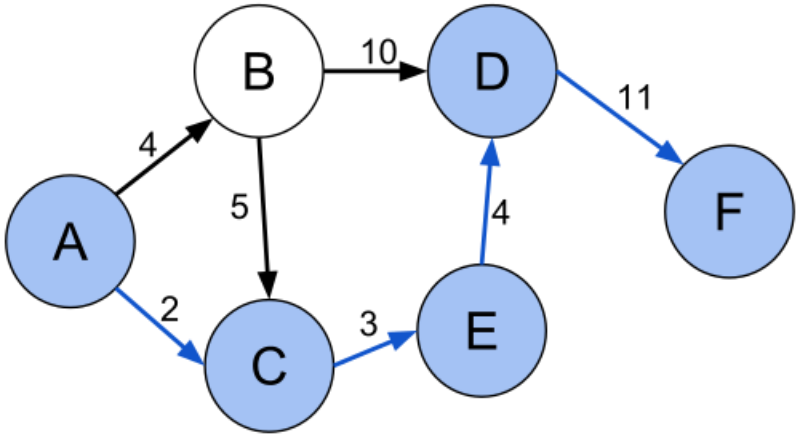
- Prim and Krustal
- Both are Greedy Algorithms
- Krustal O(E log E)
- Prim O(|E| log |V|) with binary heap and adjacency list



Graph Algorithms: Shortest Path

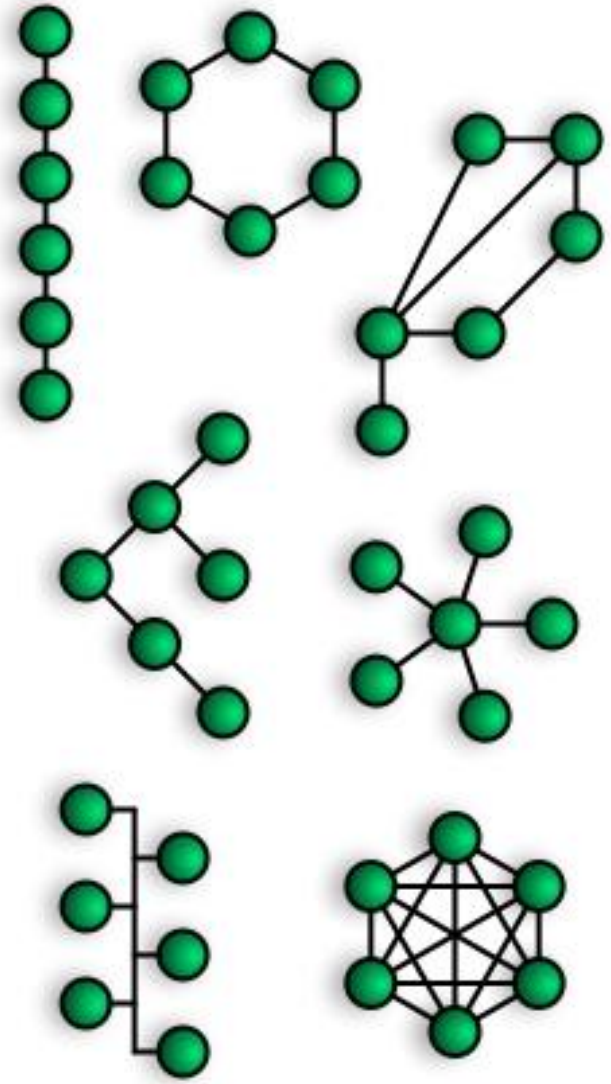
- Dijkstra's algorithm solves the single-source shortest path problem.
- Bellman–Ford algorithm solves the single-source problem if edge weights may be negative.
- A* search algorithm solves for single pair shortest path using heuristics to try to speed up the search.
- Floyd–Warshall algorithm solves all pairs shortest paths.
- Johnson's algorithm solves all pairs shortest paths, and may be faster than Floyd–Warshall on sparse graphs.

- Viterbi algorithm solves the shortest stochastic path problem with an additional probabilistic weight on each node.



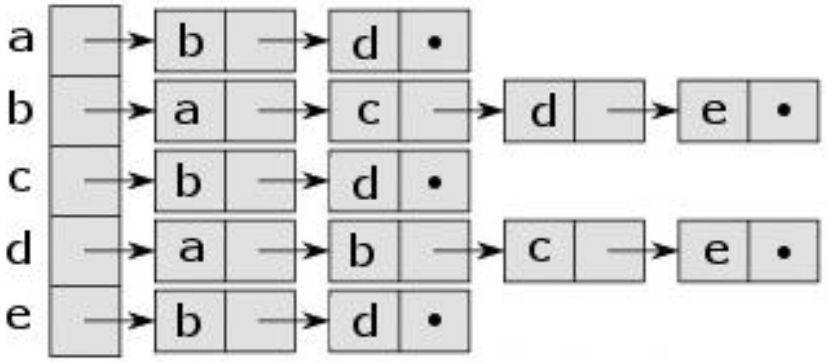
Graph Algorithms

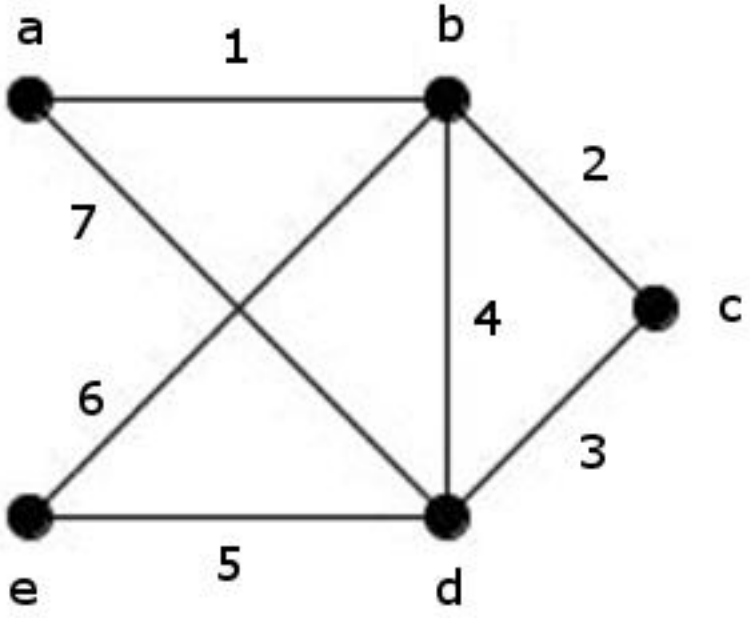
- Network Flow
  - Maximum Flow: Ford-Fulkerson algorithm
  - Min-cost Max flow problem: Edmonds-Karp
- Cycle Detection
- Shortest Cycle
- Connected Components: Union Find
- Dense Component Discovery
- Clustering



Graph Representations

- Adjacency List
- Adjacency Matrix
- Incidence Matrix





$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

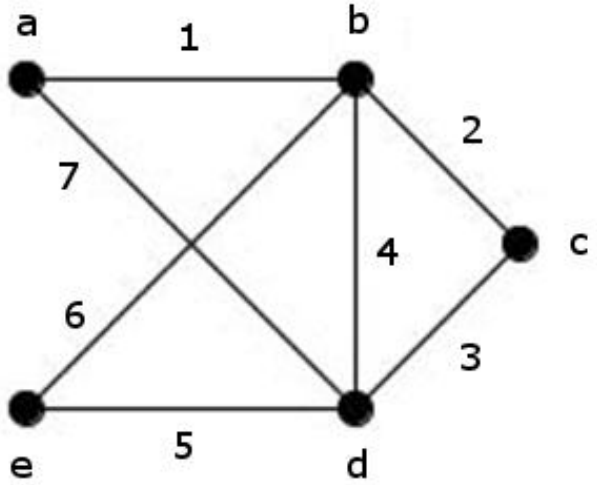
$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

Graphs and Matrices: Types

- Adjacency (A)
- Incidence (E)
- Degree (D)
- Distance (G)
- Laplacian (L)
- Signless Laplacian (L^+)

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix} \quad E = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

$$D = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix} \quad G = \begin{bmatrix} 0 & 1 & 2 & 1 & 2 \\ 1 & 0 & 1 & 1 & 1 \\ 2 & 1 & 0 & 1 & 2 \\ 1 & 1 & 1 & 0 & 1 \\ 2 & 1 & 2 & 1 & 0 \end{bmatrix}$$



$$L = \begin{bmatrix} 2 & -1 & 0 & -1 & 0 \\ -1 & 4 & -1 & -1 & -1 \\ 0 & -1 & 2 & -1 & 0 \\ -1 & -1 & -1 & 4 & -1 \\ 0 & -1 & 0 & -1 & 2 \end{bmatrix} \quad L^+ = \begin{bmatrix} 2 & 1 & 0 & 1 & 0 \\ 1 & 4 & 1 & 1 & 1 \\ 0 & 1 & 2 & 1 & 0 \\ 1 & 1 & 1 & 4 & 1 \\ 0 & 1 & 0 & 1 & 2 \end{bmatrix}$$

Graphs and Matrices: Definitions and Relationships

- Matrices are defined on conditions
- Some matrices are related to others

$$L = D - A$$

$$L^+ = D + A$$

$$L^+ = EE^T$$

$$A_{i,j} = \begin{cases} 1 & (i,j) \in E \\ 0 & \text{otherwise} \end{cases}$$

$$E_{i,j} = \begin{cases} 1 & \text{j connects to i} \\ 0 & \text{otherwise} \end{cases}$$

$$D_{i,j} = \begin{cases} d_i & i = j \\ 0 & \text{otherwise} \end{cases}$$

$$G_{i,j} = \begin{cases} & \text{d(i,j)} \end{cases}$$

$$L_{i,j} = \begin{cases} d_i & i = j \\ -1 & (i,j) \in E \\ 0 & \text{otherwise} \end{cases}$$

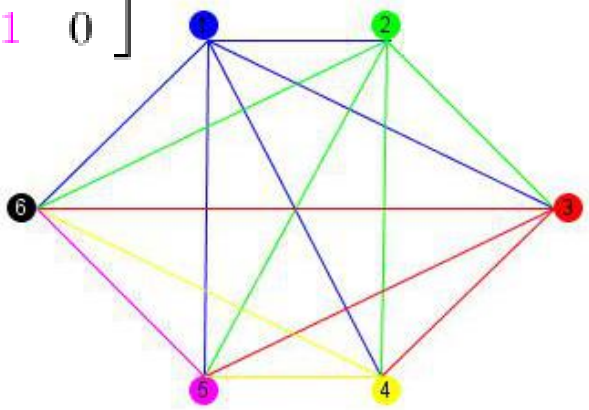
$$L^+_{i,j} = \begin{cases} d_i & i = j \\ 1 & (i,j) \in E \\ 0 & \text{otherwise} \end{cases}$$

Graphs and Matrices:
Adjacency Matrix

- Loops will be in the diagonal
- Undirected graphs have Symmetric Matrices Undirected do not
- Simple graphs only have ones. Multigraphs will have the edge count between two vertices.
- Weighted graphs will have weights

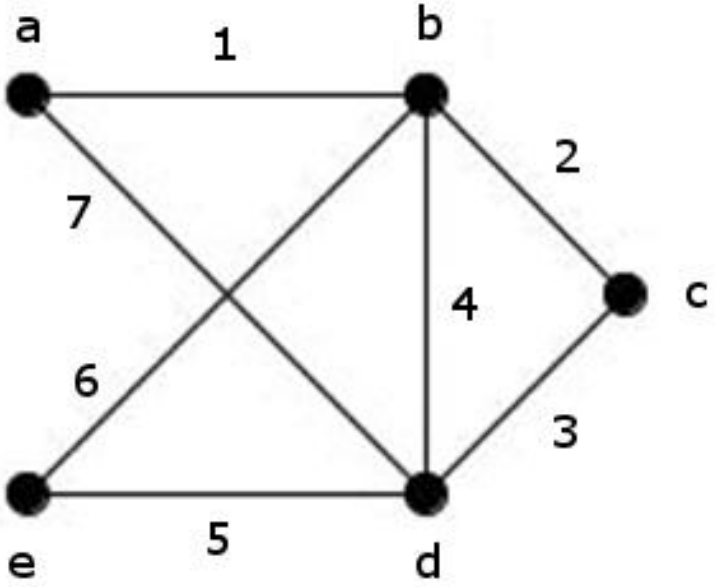
$$\begin{bmatrix} 0 & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} & a_{1,6} \\ a_{2,1} & 0 & a_{2,3} & a_{2,4} & a_{2,5} & a_{2,6} \\ a_{3,1} & a_{3,2} & 0 & a_{3,4} & a_{3,5} & a_{3,6} \\ a_{4,1} & a_{4,2} & a_{4,3} & 0 & a_{4,5} & a_{4,6} \\ a_{5,1} & a_{5,2} & a_{5,3} & a_{5,4} & 0 & a_{5,6} \\ a_{6,1} & a_{6,2} & a_{6,3} & a_{6,4} & a_{6,5} & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$



Graphs and Matrices: Walks

- Raising the Adjacency Matrix the nth power computes the number of walks between vertices



$$A = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

$$A^2 = \begin{bmatrix} 2 & 1 & 2 & 1 & 2 \\ 1 & 4 & 1 & 3 & 1 \\ 2 & 1 & 2 & 1 & 2 \\ 1 & 3 & 1 & 4 & 1 \\ 2 & 1 & 2 & 1 & 2 \end{bmatrix}$$

$$A^3 = \begin{bmatrix} 2 & 7 & 2 & 7 & 2 \\ 7 & 6 & 7 & 7 & 7 \\ 2 & 7 & 2 & 7 & 2 \\ 7 & 7 & 7 & 6 & 7 \\ 2 & 7 & 2 & 7 & 2 \end{bmatrix}$$

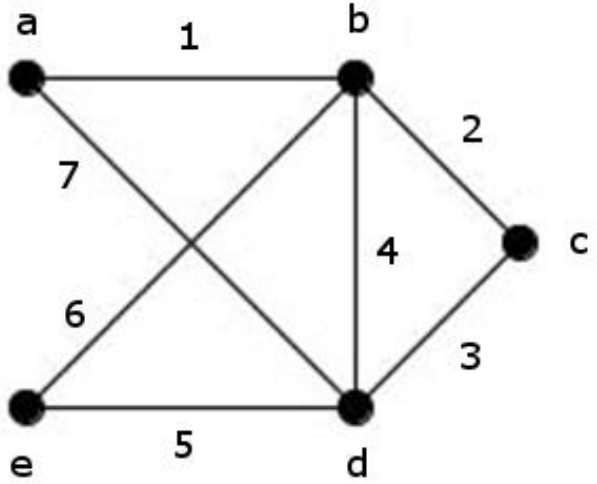
Graphs and Matrices: Spanning Trees

- Matrix tree Theorem
- Kirchhoff's theorem
- Can also be calculated using signed incidence matrix
- Robustness Measure

$$L = \begin{bmatrix} 2 & -1 & 0 & -1 & 0 \\ -1 & 4 & -1 & -1 & -1 \\ 0 & -1 & 2 & -1 & 0 \\ -1 & -1 & -1 & 4 & -1 \\ 0 & -1 & 0 & -1 & 2 \end{bmatrix}$$

$$L_{n-1} = \begin{bmatrix} 2 & -1 & 0 & -1 \\ -1 & 4 & -1 & -1 \\ 0 & -1 & 2 & -1 \\ -1 & -1 & -1 & 4 \end{bmatrix}$$

$$det\left(\begin{bmatrix} 2 & -1 & 0 & -1 \\ -1 & 4 & -1 & -1 \\ 0 & -1 & 2 & -1 \\ -1 & -1 & -1 & 4 \end{bmatrix}\right) = 20$$



$$t(G) = \frac{1}{n}\prod_{i=1}^{n-1}\lambda_i \qquad 20 = \frac{1}{5}\cdot 5\cdot 5\cdot 2\cdot 2$$


Thank you
Questions?

References and Further Reading

- Reinhard Diestel - Graph Theory
- http://diestel-graph-theory.com/
- Wikipedia: Glossary of Graph Theory
- http://en.wiktionary.org/wiki/Appendix:Glossary_of_graph_theory
- GRAPH THEORY - Keijo Ruohonen
- http://math.tut.fi/~ruohonen/GT_English.pdf
- Topics in Graph Theory (COMS 6204) - SPRING 2010
- http://www.cs.columbia.edu/~cs6204/course%20material.html
- Networks, Crowds, and Markets
- http://www.cs.cornell.edu/home/kleinber/networks-book/
- Network Analysis: Methodological Foundations
- Ulrik Brandes, Thomas Erlebach
- http://www.springer.com/computer/theoretical+computer+science/book/978-3-540-24979-5
- Wikipedia: Graph algorithms
- http://en.wikipedia.org/wiki/Category:Graph_algorithms
- CS267 Graph Algorithms Winter 2014
- http://theory.stanford.edu/~virgi/cs267/