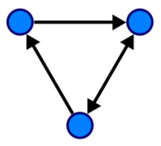
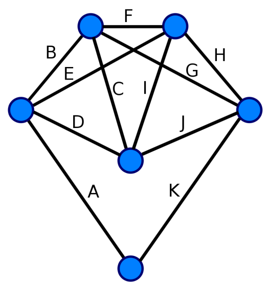
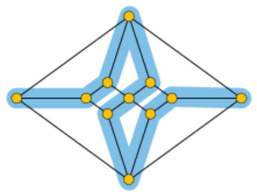
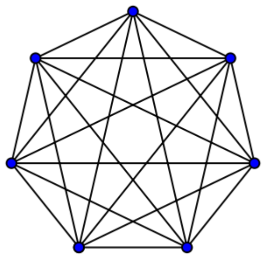
Graph	Collection of nodes connected by edges. A network that help define and visualise a relationship (edges) between components (nodes/vertex). **G = (V,E)**
	
Undirected graph	Edge (u, v) = (v,u)
Directed graph	Edge (u, v) ≠ (v, u)
Programing graphs	(1) Adjacency matrix (2) Edge list (3) Dictionary based
Walk	Sequence of connected nodes
Trail	Walk with unique edges
Path	Trail with unique nodes
Closed walk /tour/cycle	A path with the same start and end
Euler trail	An undirected graph that visits every edge exactly once. [Eulerian path - Wikipedia]
Euler cycle	A euler trail that starts and ends on the same vertex. A connected graph has an Euler cycle if and only if every vertex has even degree
	Every vertex of this graph has an even degree. Therefore, this is an Eulerian graph. Following the edges in alphabetical order gives an Eulerian circuit/cycle.
Hamiltonian path	An undirected or directed graph that visits each vertex exactly once.
Hamiltonian cycle	A cycle that visits each vertex exactly once.



Hamiltonian path - Wikipedia

Complete graph

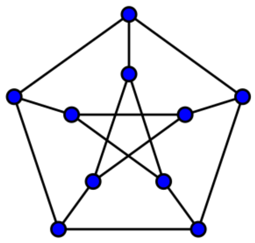
A undirected graph in which every pair of distinct vertices is connected by a unique edge



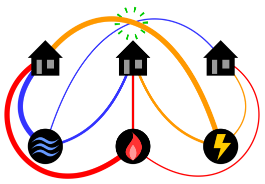
Complete graph - Wikipedia

Petersen graph

An undirected graph with 10 vertices and 15 edges.



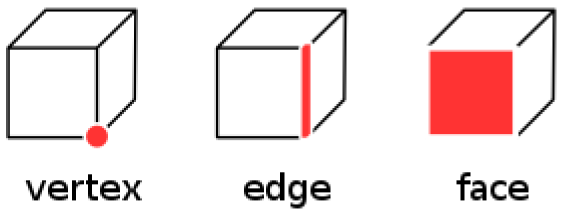
Petersen graph - Wikipedia



Why this puzzle is impossible

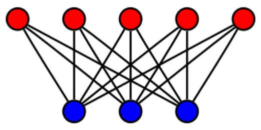
We need 9 lines to connect each facility (3) with the 3 houses. Each edge either connects to a new vertice or creates a new region. 5 edges will connect to new vertices, leaving 4 edges to create 5 separate regions. Each region costs 4 edges. There are not enough edges to satisfy this.

Euler's characteristic formula: V - E + F = 2

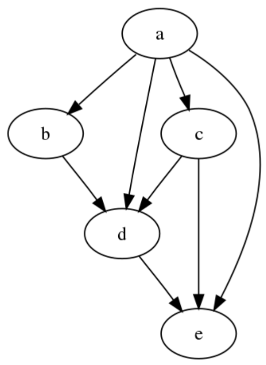
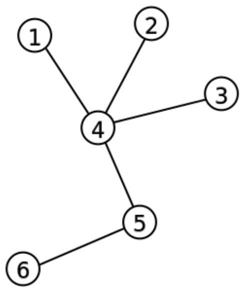


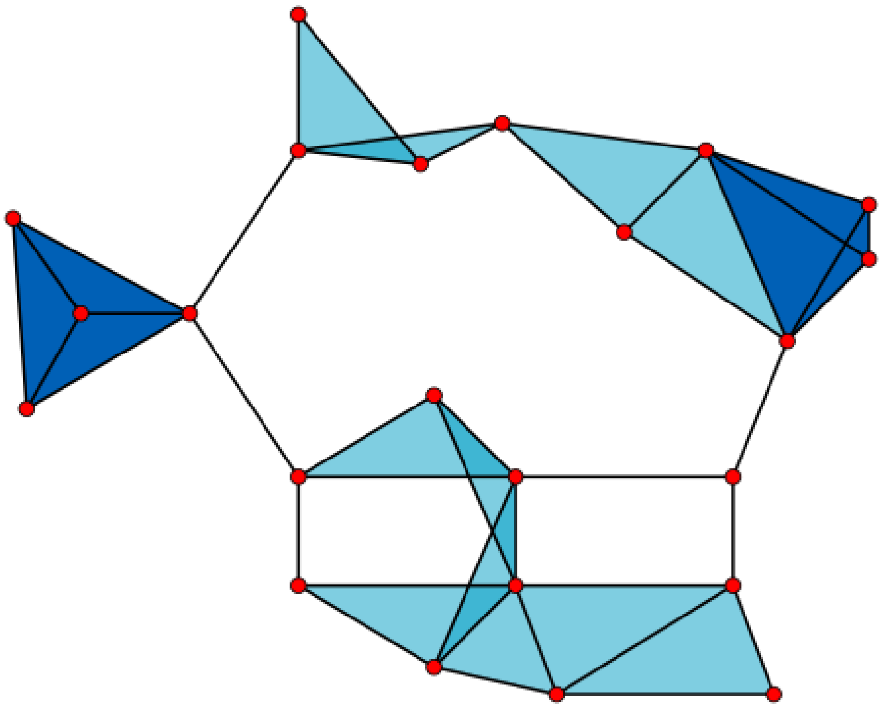
Bipartite graphs

A set of graph vertices decomposed into two disjoint sets such that edges only exist between those groups

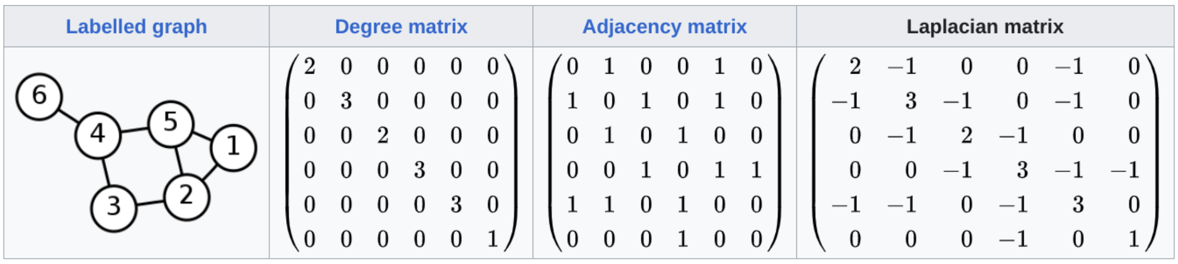


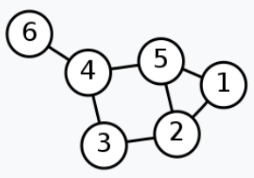
Planar graph	A graph that can be embedded into the plane i.e. it can be drawn on the plane in such a way that its edges intersect only at their endpoints (no edges cross each other).
Embedding	Nodes have no coordinates. Embedding generates N-dimensional coordinates for all nodes.
Spring layout	In a spring layout, nodes are represented as points in a two-dimensional space, and edges are represented as springs connecting pairs of nodes. The layout algorithm iteratively adjusts the positions of the nodes based on the forces exerted by the springs and other factors, aiming to achieve a balanced and visually pleasing arrangement.
Kamada-Kawai layout	graph embedding technique used to create a two-dimensional representation of a graph that emphasises the underlying distance relationships between nodes. It aims to position nodes in a way that minimises the distortion of inter-node distances while considering the connectivity of the graph.
Planar layout	A graph layout where the nodes and edges are positioned in a two-dimensional space without any edge crossings
Circular layout	In a circular layout, the nodes are positioned along a circle, and the edges are drawn as curves or straight lines connecting the nodes.
Spiral layout	The nodes of a graph are arranges in a spiral pattern.
Vertex coloring	Assign colors to nodes so that no 2 neigboring nodes have the same color
Chromatic number	Minimum number of colors needed to achieve vertex coloring. For planar graphs: four colour theorem - any planar graph can be colored using at most four colors in such a way that no two adjacent nodes share the same color.
Indegree (v)	Number of edges pointing to node v in a directed graph
Outdegree (v)	Number of edges leaving node v
Directed acyclic graph (DAG)	A directed graph with no directed cycles. Each edge is directed from one vertex to another such that following those directions will never form a closed loop.

	A directed graph is a DAG only if it can be topologically ordered Examples: dependency graphs, family trees, citation networks, scheduling with ordering constraints Directed acyclic graph - Wikipedia
Dependency graph	A directed graph representing dependencies of several objects towards each other. Dependency graphs without circular dependencies form DAGs.
Trees 	An undirected graph in which any two vertices are connected by exactly one path. It is a connected, acyclic, undirected graph. Tree (graph theory) - Wikipedia
Reachability	The ability to get from one vertex to another within a graph. Strongly connected: every node is reachable from all other nodes Weakly connected: THe underlying undirected graph is connected Reachability - Wikipedia
Topological sorting	A linear ordering of vertices such that every directed edge *uv* from vertex *u* to vertex *v* comes before *v* in the ordering. Topological sorting is possible only if the graph has no directed cycles (in other words, only if the graph is a DAG). Topological sorting - Wikipedia.
Breadth first search (BFS) algorithm	Algorithm that is used to search a graph data structure for a node that meets a set of criteria. It starts at the root of the graph and visits all nodes at the current depth level before moving on to the nodes at the next depth level.
Khan's algorithm	Keeps track of the number of incoming edgesinto each node (indegree). It repeatedly: finds nodes with no incoming edge, that is, nodes with zero indegree (no dependency)
Communities	Groups of nodes that are closely connected.

Reinforcing loops	Neighbours update neighbours
Cliques	An induces subgraph of the graph that is complete *Let G = (V,E) be any graph, and let S ⊂ V be any subset of vertices of G. Then the induced subgraph G[S] is the graph whose vertex set is S and whose edge set consists of all of the edges in E that have both endpoints in S. That is, for any two vertices (u,v) ∈ S, u and v are adjacent in G[S] if and only if they are adjacent in G* 
Clustering coefficient	The degree to which neighbours of v are also neighbours of each other. **Definition 6.8:** *Consider a simple connected, undirected graph G and vertex $v \in V(G)$ with neighbor set $N(v)$. Let $n_v = \lvert N(v)\rvert$ and m_v be the number of edges in the subgraph induced by $N(v)$, i.e., $m_v = \lvert E(G[N(v)])\rvert$. The* ***clustering coefficient*** *$cc(v)$ for vertex v with degree $\delta(v)$ is defined as* $cc(v) \overset{\text{def}}{=} \begin{cases} m_v / \binom{n_v}{2} = \frac{2 \cdot m_v}{n_v(n_v-1)} & \textit{if } \delta(v) > 1 \\ \textit{undefined} & \textit{otherwise} \end{cases}$
Centrality	Which nodes are in the middle of the graph
Eccentricity	**Definition 6.6:** *Consider a connected graph G and let $d(u,v)$ denote the distance between vertices u and v. The* ***eccentricity*** *$\epsilon(u)$ of a vertex u in G is defined as* $\max\{d(u,v) \mid v \in V(G)\}$. *The* ***radius*** *$rad(G)$ is equal to* $\min\{\epsilon(u) \mid u \in V(G)\}$. *Finally, the* ***diameter*** *of G is the maximal shortest path between any two vertices:* $diam(G) = \max\{d(u,v) \mid u,v \in V(G)\}$.
Vertex centrality	1 / eccentricity

Adjacency matrix	Matrix to describe a graph. 0=no edge, 1 = edge. $\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$
Graph Laplacian	L = D - A Where D is the degree matrix and A is the adjacency matrix of the graph.



Labelled graph	Degree matrix	Adjacency matrix	Laplacian matrix
	$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$

Spectral graph theory	The study of the properties of a graph in relationship to the characteristic polynomials, eigenvalues and eigenvectors.
Spectral embedding	Spectral embedding is a dimensionality reduction technique commonly used in machine learning and data analysis tasks. It is particularly useful for visualising high-dimensional data in a lower-dimensional space. The main idea behind spectral embedding is to leverage the eigenvalues and eigenvectors of a similarity matrix or graph Laplacian matrix derived from the data.
Graph partitioning	Graph partitioning refers to the process of dividing a graph into disjoint subsets or partitions. The goal is to group the nodes of the graph such that nodes within the same partition have stronger connections among themselves compared to nodes in different partitions. It is a fundamental problem in graph theory and has numerous applications in various domains, including computer science, data analysis, network optimization, and parallel computing.
Spectral graph partitioning	Spectral graph partitioning is a technique for dividing a graph into multiple partitions based on spectral properties. It leverages the eigenvalues and eigenvectors of a graph's Laplacian matrix to guide the partitioning process. The Laplacian matrix captures the relationships between nodes in a graph and plays a crucial role in spectral graph theory.

Markov modelling

https://github.com/anoekkooijmans/Graph-theory.git

In the context of graph theory, Markov modeling can be used to study and analyze processes that occur on graphs. It involves applying Markov chains to model the probabilistic transitions between different states or nodes in a graph.

Here's how Markov modeling is applied to graphs:

Graph representation: The first step is to represent the system or process of interest as a graph. The nodes of the graph represent states or entities, while the edges represent the transitions or connections between these states. The graph can be directed or undirected, and the edges can have weights or probabilities associated with them.
State assignment: Each node in the graph is assigned a state, representing a specific condition, configuration, or characteristic of the system being modeled. The states can be discrete (e.g., labels or categories) or continuous (e.g., numerical values).
Transition probabilities: The edges of the graph are assigned transition probabilities, indicating the likelihood of transitioning from one state (node) to another. These probabilities can be determined based on empirical data, expert knowledge, or other sources of information.
Transition matrix: A transition matrix is constructed based on the graph representation. The rows and columns of the matrix correspond to the states (nodes) in the graph, and each element of the matrix represents the probability of transitioning from one state to another. The matrix can be sparse if there are many zero probabilities.
Markov chain analysis: Once the transition matrix is obtained, standard techniques from Markov chain theory can be applied to analyze the dynamics of the system. This includes studying the long-term behavior, calculating steady-state probabilities or stationary distributions, and analyzing various properties of the Markov chain, such as ergodicity or irreducibility.

By applying Markov modeling to graph theory, it becomes possible to study processes that evolve on graphs and understand how the states or entities in the graph transition over time. This approach finds applications in various domains, such as modeling random walks on networks, analyzing information spreading or epidemic processes on graphs, predicting the evolution of social networks, and studying diffusion processes in complex systems.

Overall, Markov modeling provides a powerful framework for understanding and characterizing the dynamics of systems on graphs, leveraging the principles of probability and stochastic processes.