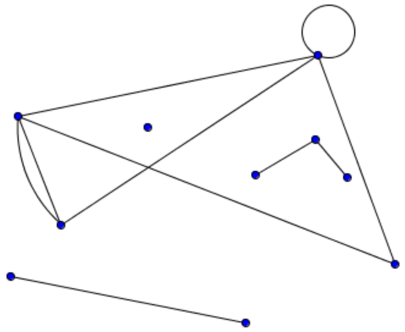
# Graph Theory Terminology

**Graph**

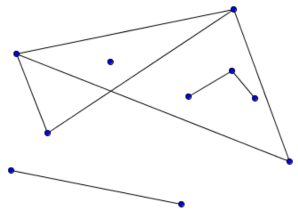
A collection of edges and vertices.
A *vertex* is a point (also called a node) which forms a junction between two edges.
An *edge* is a line (also called an arc) which connects two vertices.
The *valency* or *order* or *degree* of a vertex is the number of edges connected to it.



A **network** is a graph with weighted edges, and a **digraph** is a graph with directed edges.
A **bipartite graph** is one where vertices from one subset only link to those in the other.

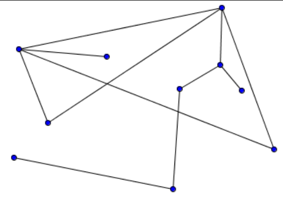
**Simple Graph**

A graph which contains no *loops* (an edge from a vertex to itself) and no *duplicate edges* (more than one edge linking two vertices)



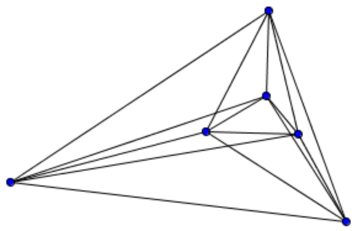
**Connected Graph**

A graph in which every pair of vertices is *connected* (ie, there exists an unbroken route between them).



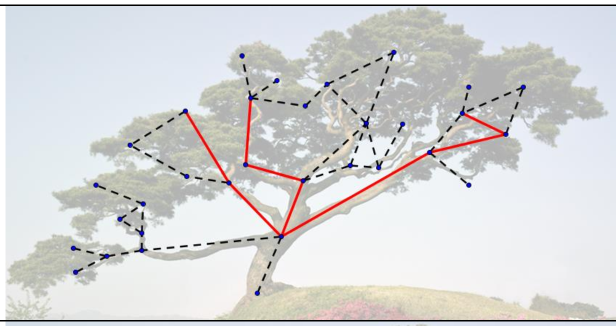
**Complete Graph**

A *simple graph* where every pair of vertices is linked by an edge.
The complete graph with $n$ vertices ($K_n$) has $\frac{n(n-1)}{2}$ edges.

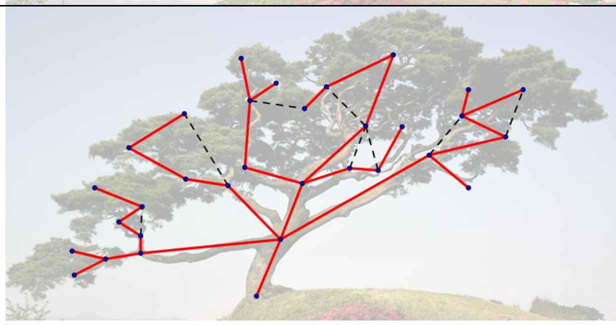


**Tree**

A connected graph (eg a subgraph) with no cycles.



**Spanning Tree**
A *tree* that connects all vertices. Any connected graph has a spanning tree, and a graph with $n$ vertices will have $n - 1$ edges in its spanning tree.
Note: a *minimum spanning tree*, for a network, is a spanning tree with the smallest possible total weighting of edges.



**Kruskal's Algorithm** finds a minimum spanning tree by selecting edges in order of weighting, smallest first, ensuring no cycles are created as each edge is added.

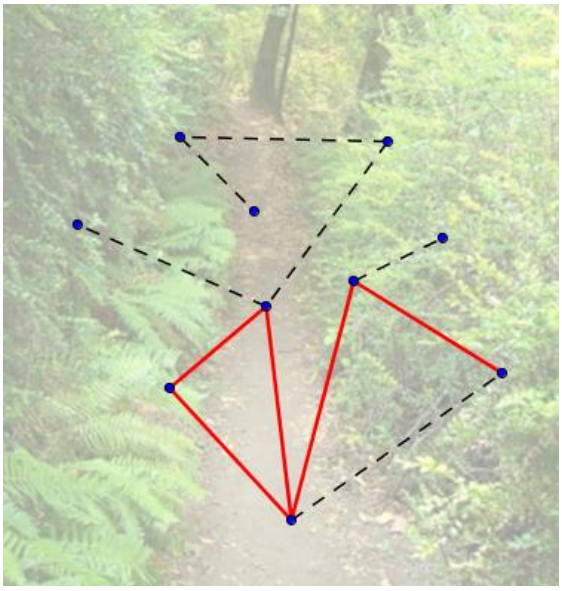
**Prim's Algorithm**, which can also be applied easily in matrix form, finds a minimum spanning tree by starting from a given vertex and at each step linking to the nearest unused vertex.

**Trail**

A walk (a route through the graph) that doesn't repeat edges.

An *Eulerian Trail* traverses every edge (and, since it is a trail, that means exactly once).

A graph with an Eulerian trail starting and finishing at the same vertex is *Eulerian* (if the Eulerian trail starts and finishes at different vertices the graph is *semi-Eulerian*). Eulerian graphs have no odd vertices, and semi-Eulerian graphs have exactly two odd vertices.



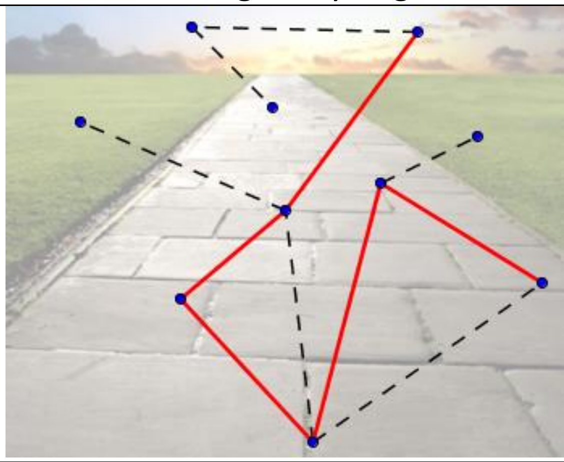
The **Chinese Postman** algorithm augments a network by adding edges to make it Eulerian or semi-Eulerian in order to traverse the minimum distance while covering every edge.

**Path**

A *trail* that doesn't repeat vertices (except sometimes first and last).

Note: If you can't repeat vertices, it is impossible to repeat edges, so a path is necessarily also a trail.

A *Hamiltonian path* visits every vertex (and, since it is a path, that means exactly once).

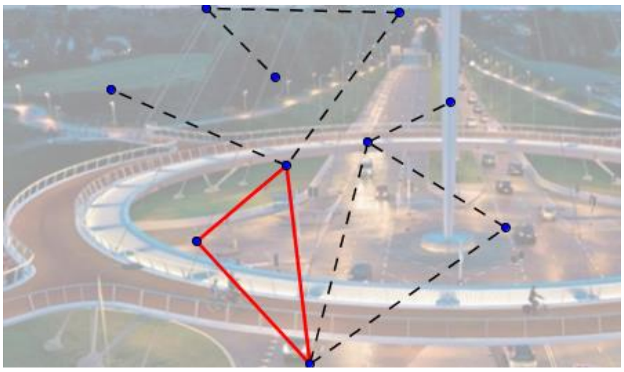


**Cycle**

A *path* whose first and last vertices are the same (a closed path with at least one edge).

Note: since it is a path it cannot repeat vertices or edges. Also, it must contain at least two vertices (otherwise it would be a loop).

A *Hamiltonian cycle* visits every vertex (and, since it is a cycle, that means exactly once, but starting and finishing at the same vertex). Also called a '*tour*'.



The **Travelling Salesman Problem** attempts to find a Hamiltonian cycle (a tour) with the minimum weighting for a network. By combining minimum spanning trees with two additional edges (the **lower bound algorithm**), lower bounds are found, and by finding examples of Hamiltonian cycles (the **nearest neighbour algorithm**), upper bounds are found. A direct solution is rarely computationally feasible.