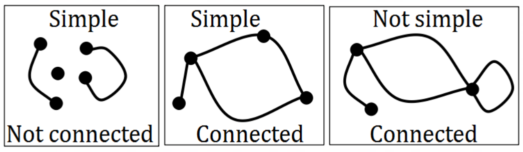
Graph. A graph G is a set of vertices and edges that represent how objects are related.

Edge. An edge joins two adjacent vertices. Two or more edges which connect the same pair of vertices are called multiple edges. Vertices may (less strictly) be called *nodes*, faces *regions* and edges *arcs*, though some texts only use *arc* for directed graphs.

Loop. A loop is an edge that joins a vertex to itself.

Degree of vertex. The degree of a vertex is the number of edges that meet at the vertex (note that both ends of a loop count). Degree of vertex will be even or odd. For any graph, degree sum is twice number of edges and so is always even. Odd vertices always occur in pairs.

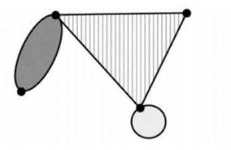
Simple graph. A simple graph has no loops or multiple edges. Can be connected or disconnected and usually unweighted and undirected. Connected graph cannot be simple if its highest degree is greater than or equal to its number of vertices.



Connected graph. A graph is connected if a path exists between all pairs of vertices. (*Otherwise, disconnected*).

Bridge. A bridge is an edge whose removal increases the number of components of a graph. If graph is connected, removal leaves graph disconnected.

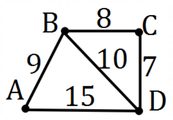
Planar graph. A planar graph is a graph that ***can*** be drawn in the plane. Does **not** need to be connected. A plane graph is a planar graph drawn in the plane so that no edges cross. (*Otherwise non-planar*). A plane graph will have at least one face. Outer region also counts as a face. The graph shown has $v = 4, e = 6$ and $f = 4$.



Euler's rule / formula: $v + f - e = 2$. Only applies to connected planar graphs.

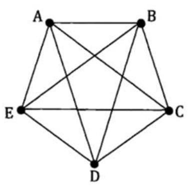
Directed graph or digraph. Some, or all, edges are directed with arrows. A directed edge is called an arc. (*Otherwise undirected, non-directed or two-way*).

Network or weighted graph. Network is a graph in which each edge is labelled with a number used to represent distance, time, cost, etc.



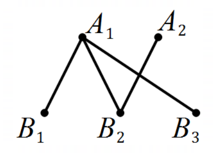
Subgraph. When vertices and edges of G_A are also vertices and edges of G_B, G_A is a subgraph of G_B.

Complete graph. A complete graph K_n is a simple graph in which all n vertices are joined to all other vertices by an edge. A complete bipartite graph is a bipartite graph where every vertex of the first set is connected to every vertex of the second set. K_5 is shown.



Complement of a graph. To draw the complement of a graph, add all the missing edges required to form a complete graph, and then remove all the original edges.

Bipartite graph. A bipartite graph is a graph whose set of vertices can be split into two distinct groups in such a way that each edge of the graph joins a vertex in the first group to a vertex in the second group. Also see complete graph.



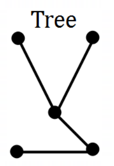
Walk. A walk in a graph is a sequence of vertices such that from each of its vertices there is an edge to the next vertex in the sequence. Can include repeated edges or vertices (not necessarily distinct).

Trail. A trail is a walk with distinct (no repeat) edges.

Path. A path is a walk such that all the vertices and edges are distinct (different).

Open / closed. A walk that starts and finishes at different vertices is said to be open, whilst one that starts and finishes at the same vertex is said to be closed.

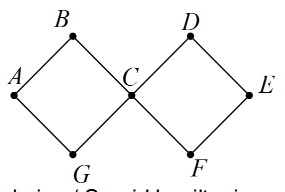
Cycle. A cycle is a closed path with at least one edge in which only the first and last vertices are equal. *(Do not confuse with a circuit, which is different and not in syllabus)*.



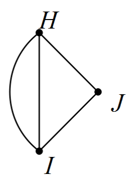
Tree. A tree is a connected graph with no cycles.

Length. The length of a walk is the number of edges it includes.

Eulerian trail / graph. Eulerian trail visits every edge in a connected graph once only, but it may include repeated vertices. If this trail is closed the graph is Eulerian, whilst if it is open the graph is semi-Eulerian. Closed trail will exist if all vertices of graph are even. Open trail will exist if graph has exactly 2 odd vertices (will start at one odd vertex and finish at the other). Connected graph does not have to be planar.
If Eulerian trail exists, graph can be referred to as traversable.



Eulerian / Semi-Hamiltonian



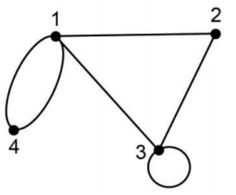
Hamiltonian / Semi-Eulerian

Hamiltonian and semi-Hamiltonian graphs.
A graph is Hamiltonian if it contains a cycle (closed path) that includes all vertices once.
A graph is semi-Hamiltonian if it contains a path that includes all vertices once but does not contain a Hamilton cycle.
A Hamiltonian graph can always be converted to a semi-Hamiltonian graph by removing one edge from the cycle.

Distance matrix. In a distance matrix, the numbers give the distance between each pair of vertices.

Adjacency matrix
An adjacency matrix for a non-directed graph with n vertices is an $n \times n$ matrix in which the entry in row i and column j is the number of edges joining the vertices i and j. In an adjacency matrix, a loop is counted as 1 edge. Sum of matrix coefficients plus sum of leading diagonal is twice number of edges.

Non-directed graph

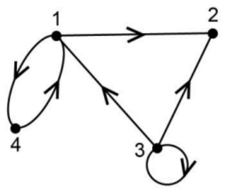


Adjacency matrix

$$\begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 1 & 1 & 2 \\ 2 & 1 & 0 & 1 & 0 \\ 3 & 1 & 1 & 1 & 0 \\ 4 & 2 & 0 & 0 & 0 \end{array}$$

For a directed graph the entry in row i and column j is the number of directed edges (arcs) joining the vertex i and j in the direction i to j. Sum of coefficients is number of edges.

Directed graph



Adjacency matrix

$$\begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 1 & 0 & 1 \\ 2 & 0 & 0 & 0 & 0 \\ 3 & 1 & 1 & 1 & 0 \\ 4 & 1 & 0 & 0 & 0 \end{array}$$

Squares or cubes of adjacency matrix are used to determine the number of 2 or 3 step routes that exist.

Assignment problems/Hungarian algorithm.
Use inspection for small scale problems, or Hungarian algorithm to minimise assignment using $n \times n$ table.
Algorithm steps:

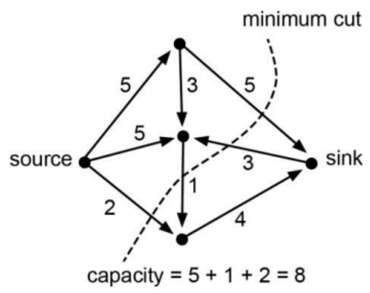
- *If required to maximise assignment, start by subtracting all entries from largest.*
- *May also need to add dummy row or column of zeroes to make table square.*

1. Subtract the smallest entry in each row from all the entries in that row.
2. Subtract the smallest entry in each column from all the entries in that column.
3. Cancel out all zeros with minimum number of horizontal/vertical lines.
4. If the number of lines is n, skip to 7.
5. Determine m, the smallest number not crossed out. Subtract m from all numbers not crossed out. Add m to all numbers where zero lines intersect.
6. Remove lines and go back to 3.
7. Make assignment using zeros.

Go to *http://www.hungarianalgorithm.com* to learn more.

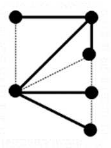
Shortest path. Syllabus says use trial-and-error to determine the shortest path between two vertices in a weighted graph. Can also use Dijkstra's algorithm.

Maximum flow problems. Source: vertex has all flows outwards. Sink: vertex has all flows inwards. Cuts pass through edges and separate source from sink. Value of cut is sum of flows through edges that are cut. Flows across cut from sink to source are 0. Value of minimum cut is the maximum flow (aka maximum flow–minimum cut theorem).



Alternative: list paths from source to sink and find largest amount that can flow along each path, reducing flows as you go. Add up flows to obtain maximum. Minimum cut will pass through edges that all have zero spare capacity.

Spanning tree. A spanning tree is a subgraph of a connected graph that connects all vertices and is a tree.



Minimum spanning tree. Use to solve minimal connector problems. For a weighted graph, it is the spanning tree with minimum sum of weights. Use Prim's algorithm with diagram or table: Imagine graph has no edges. Start at any vertex and add bridge to nearest unconnected vertex. From either of these vertices, add bridge to the nearest unconnected vertex and so on. End with a tree.

Project Networks. Digraph consisting of a sequence of connected tasks where the next task cannot start until all immediate predecessors have been completed. Minimum completion time is the least time to complete all tasks in order. Use forward scan (start to finish) (max) to find earliest start time (EST) for each task. Use backward scan (min) to find latest finish time (LFT) for each task. Latest start time (LST) is LFT less duration (D). Float (F), or slack time, is LST less EST.

$$LST = LFT - D, \qquad F = LST - EST = LFT - D - EST.$$

The path that gives minimum completion time is the critical path. Delays on critical path will delay completion of the project. To find number of people required to complete project in minimum time, assign one to critical path and then add others until all tasks are covered.

Travelling Salesman Problem. This can be modelled using an undirected weighted graph, such that cities are the vertices, connecting roads are the edges, and distances are the edge weights. The problem is to start and finish at a specified vertex after having visited each other vertex exactly once, minimising the distance travelled. Use trial-and-error to determine Hamilton cycle of minimum length.

Round-robin sporting competition. A single round-robin sporting competition is a competition in which each competitor plays each other competitor once only.

Food web. A food web (or chain) depicts feeding links (who eats whom) in an ecological community.