Algorithms from Decision 1

Algorithm	Purpose	Summary	Examples
Kruskal's algorithm	To find a minimum spanning tree for a network.	Select edges in order of lowest weighting.	Laying cable for an electricity, phone or internet connection.
Prim's algorithm	To find a minimum spanning tree for a network.	Connect vertices by adding the lowest weighted edge each time.	Laying cable for an electricity, phone or internet connection.
Dijkstra's algorithm	To find the shortest path through a network.	Use successive labelling to find the shortest path to each vertex.	Route planning (eg Sat Nav), Internet traffic routing.
Chinese postman algorithm	To find the shortest path which traverses every edge of a network.	Eliminate odd vertices by adding new edges, then find an Eulerian trail traversing the new network.	Postman delivering to every house along every road, bin-men's route planning, sight-seeing.
Nearest-neighbour algorithm	To find an upper bound for the shortest path which links every vertex of a network.	Beginning at a chosen vertex, choose the shortest edge each time until all vertices have been visited.	A travelling salesman, a delivery van which needs to make multiple stops.
Lower bound algorithm	To find a lower bound for the shortest path which links every vertex of a network.	Delete a vertex and connected edges, and find a minimum spanning tree for the remaining graph	A travelling salesman, a delivery van which needs to make multiple stops.
Alternating path algorithm	To find a maximal matching between two sets.	Beginning with an unconnected vertex, connect it, deleting edges and connecting vertices from alternate sides.	Speed dating, making arrangements for seating plans at a wedding, matching up workers with jobs or companies with clients.
Bubble sort algorithm	To order an unordered list.	Compare, and, if needed, swap successive pairs of items. Repeat until done.	Computerised ordering of lists (of numbers or other data), for example to facilitate rapid data look up.
Shuttle sort algorithm	To order an unordered list.	Compare first two items, and swap if needed. Introduce the next item, and insert it into the list where needed. Repeat.	Computerised ordering of lists (of numbers or other data), for example to facilitate rapid data look up.
Shell sort algorithm	To order an unordered list.	Split the data into sublists, shuttle sort each separately, then combine sublists and repeat.	Computerised ordering of lists (of numbers or other data), for example to facilitate rapid data look up.
Quick sort algorithm	To order an unordered list.	Select a pivot item and compare each subsequent item to it, creating sublists to either side. Repeat with each sublist.	Computerised ordering of lists (of numbers or other data), for example to facilitate rapid data look up.

Algorithm etiquette:

A well-formulated algorithm will have the following properties:

- Finite number of instructions
- Precisely defined stages
- Precise instructions
- Answer must depend only on the input variables
- Algorithms must work (produce a result) for any valid input

When presented as a flow chart:

- Oval boxes are for starting and stopping, and for inputting and outputting data.
- Square boxes are for calculations or instructions.
- Diamond boxes are for decisions.