Topic 9: Networks/Graphs/Optimal Paths

1) Terminology:

Key Terms Networks/Graphs:

1) A graph is made up of points called nodes, or vertices, that are joined together with lines known as edges or arcs.
2) The degree, valency or order of a vertex is the number of edges meeting at that vertex.
3) Sometimes edges can be assigned a value known as a weight. The weights could be distances or costs.
4) Sometimes the arcs can have a direction. If they do have arrows, then the graph is known as a directed graph, or digraph.
5) A walk of length n is a succession of n edges where the end of one is the start of the next. Examples in the diagram below would be ACDF (a walk of length 3) or ABCEF (a walk of length 4)
6) A walk in which all the nodes are different is known as a path. An example in the diagram above would be ACDF. An example of a walk that wouldn't be a path would be ACBECDF as node C is visited twice.
7) A cycle is a path that starts and ends at the same vertex.

The Hand-shaking Lemma: The degree total of a graph is double the number of edges in the graph.

2) Matrices:

a) Adding/Subtracting Matrices:

Adding 2x2 Matrices

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} + \begin{pmatrix} e & f \\ g & h \end{pmatrix} = \begin{pmatrix} a+e & b+f \\ c+g & d+h \end{pmatrix}$$

Adding 3x3 Matrices

$$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} + \begin{pmatrix} p & q & r \\ s & t & u \\ v & w & x \end{pmatrix} = \begin{pmatrix} a+p & b+q & c+r \\ d+s & e+t & f+u \\ g+v & h+w & i+x \end{pmatrix}$$

b) Multiplying Matrices:

Multiplying 2x2 Matrices

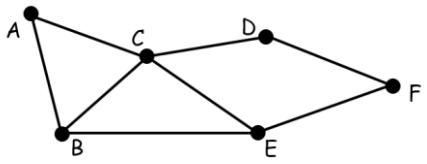
$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} e & f \\ g & h \end{pmatrix} = \begin{pmatrix} ae+bg & af+bh \\ ce+dg & cf+dh \end{pmatrix}$$

Multiplying 3x3 Matrices

$$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \begin{pmatrix} p & q & r \\ s & t & u \\ v & w & x \end{pmatrix} = \begin{pmatrix} ap+bs+cv & aq+bt+cw & ar+bu+cx \\ dp+es+fv & dq+et+fw & dr+eu+fx \\ gp+hs+iv & gq+ht+iw & gr+hu+ix \end{pmatrix}$$

c) Adjacency Matrices:

- An adjacency matrix uses figures to represent the number of edges connecting different nodes together.
- When reading matrices, we read them left to top (Lawn Tennis). E.g. in the matrix shown below, the second row represents all the connections to node B.
- The adjacency matrix for the graph shown below is shown on the right:
- Adjacency for Digraphs may not be symmetrical.



	A	B	C	D	E	F
A	0	1	1	0	0	0
B	1	0	1	0	1	0
C	1	1	0	1	1	0
D	0	0	1	0	0	1
E	0	1	1	0	0	1
F	0	0	0	1	1	0

If M is an Adjacency Matrix, then M^n gives the number of walks of length n between any two nodes in the network.

3) Trees & Minimum Spanning Trees:

a) Terminology:

- A tree is a connected graph with no cycles.
- A spanning tree for a graph P is a subgraph of P, which contains all the vertices of P and is a tree. For example, a graph is shown below on the left with three possible spanning trees alongside.
- A minimum spanning tree of a weighted graph is the spanning tree such that the total weight is a minimum.

a) Kruskal's Algorithm:

Key Step:

- Pick the smallest edges in order, taking care not to create any cycles.

c) Prim's Algorithm:

Key Steps:

1. Start at ANY node.
2. Choose the smallest weight of all nodes to the first node chosen.
3. Choose the next smallest weight that brings in another node, without creating any cycles.

Note: Have to be able to use Prim's on a Distance Matrix (weights in table form)

4) Optimal Paths:

a) Dijkstra's Algorithm

- Used to find the shortest path between any two nodes in a network
- Label each node as follows:

Node	Order of Labelling	Final Value
Working Values		

Steps:

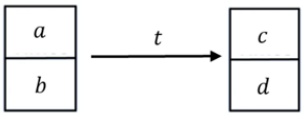
1. Start at source node and label as S, 1, 0.
2. Update temporary working values for any nodes connected to S.
3. Choose smallest weight and make labels permanent.
4. Continue until sink node is visited.

b) Critical Paths/Activities:

- The process starts at the source node and ends at the sink node.
- A precedence table, which lists all the activities that have to be done, and which ones are dependent on previous ones.
- The early time of an event is the earliest time that all preceding events are finished.
- The late time of an event is the latest time by which all preceding events have to be completed to prevent delaying the project.
- The total float of an activity is the time the start of a project could be delayed, without delaying the entire project.
- A critical path is a path from source node to sink node which connects only the critical activities together.

Steps to find critical path/activities:

1. Make out a precedence table.
2. Draw an Activity Network from the table, using dummy activities where needed.
3. Move forwards through the network calculating early times. (For two with the same value……Early = Enormousest)
4. Move in reverse through the network calculating late times. (For two with the same value……Late = Least)
5. Calculate Total Float for each activity using:



$$\text{Total Float} = d - a - t$$

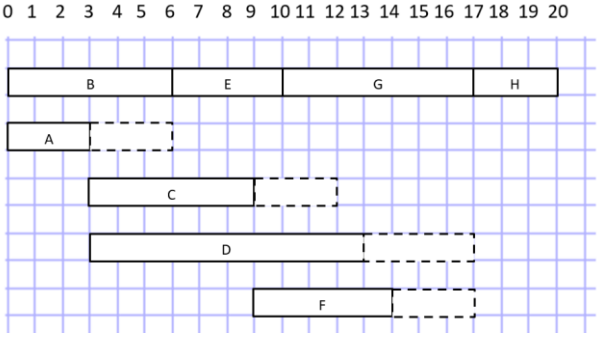
6. Write down the critical activities i.e. ones that have a total float of 0.
7. Write down the critical paths i.e. any paths from source node to sink node which connect critical activities together.

c) Scheduling:

- A Gantt Chart is a way of displaying an activity network, showing the time taken to complete activities and the total float times of each activity.

Steps for scheduling:

1. Draw Activity Network and work out all critical activities.
2. Display the timing of all activities on a Gantt Chart as shown below:



To complete in critical time:

3a. To complete the project in critical time, drop a line through the Gantt Chart and check for intersecting boxes, to see how many workers required.

3b. Assign to workers in order to minimise idle time, as much as possible. If there is a choice of 1 or more activities, assign earliest Late Times first and in alphabetical order after that.

To complete with lower numbers of workers:

3a. Calculate the lower bound using:

$$\text{Lower Bound} = \frac{sum\ of\ all\ activity\ times}{critical\ time\ for\ the\ project}$$

3b. Assign to workers in order to minimise idle time, as much as possible. If there is a choice of 1 or more activities, assign earliest Late Times first and in alphabetical order after that.

3c. Take care at the end how you assign the last few activities, to minimise the time required.

d) Bellman's Principle

- Used to find the optimal route through an activity network.
- Bellman's Principle of Optimality: Any part of an optimal path is itself optimal.
- A stage is a collection of directed edges, which link the set of nodes which are n moves from B to those that are $n-1$ moves from B. Stages are counted backwards from B.
- Each node in a multi-stage network is called a state.
- Each directed edge in a multi-stage is called an action. The weight of that edge denotes the outcome of that action, which could be time, distances or costs, for example.

Steps:

1. Draw a vertical table with columns for Stage, State, Action and Value.
2. Always start at the sink state (T) and work backwards in stages.
3. Find the optimal value for each of the states at the start of that stage.
4. Mark the optimal value with a asterisk *.
5. Keep working backwards through the stages, until the optimal path from source to sink is found.
6. At each stage, you use the asterisked value from the previous stage, according to Bellman's Principle of Optimality.

Read off the optimal path using the asterisks.

e) Dynamic Programming

- 4 types of problem:
  - Routing
  - Stock Control
  - Resource Allocation
  - Equipment Maintenance

Setup for Tables for each Type:

Routing:

Stage	State	Action	Destination	Value
# weeks to go	Where Sam visits	Where she travels	Where she finishes	= Profit – Travel + OVD

Stock Control:

Stage (Demand)	State	Action	Destination	Value
The month (No. in order book)	No. in Stock	No. Made	No. Left in Stock	Costs = Storage + Overheads + Hired Labour + OVD

Resource Allocation:

Stage	State	Action	Destination	Value
(Product)	(Tonnes available)	(Tonnes allocated)	(Tonnes remaining)	= profit + OVD

Equipment Maintenance:

Stage	State	Action	Destination	Value (Cost)
(Year)	(No. of yrs left when you buy this van)	(Decide to keep van for his no. of years)	(No. of yrs left when you sell the van)	= Cost of a new van + maintenance – resale value + OVD
0				