MEI

Mathematics in Education and Industry

MEI Structured Mathematics

Module Summary Sheets

Decision Mathematics 1
(Version B: reference to new book)

Topic 1: Algorithms

Topic 2: Graphs

Topic 3: Networks

Topic 4: Critical Path Analysis

Topic 5: Linear Programming

Topic 6: Simulation

MEI, Albion House, Market Place, Westbury. BA13 3DE
Tel: 01373 824343.   Fax: 01373 824820

References:
Chapter 1
Pages 1-7

Exercise 1B
Q. 2

Terminology

An *Algorithm* is a set of instructions for carrying out a task. There are many ways in which an algorithm can be expressed, including:

- Written language
- Pseudo-code (an abbreviated form of language, putting on instruction per line and making use of specific words such as repeat)
- Flow chart
- Computer program

Order or Complexity – the order of an algorithm is a measure of the amount of work (number of operations, instructions) needed to complete it. It is expressed in terms of n, where n is the size of the problem. It is a proportion, not an exact measure. $O(n^2)$ means that the algorithm takes an amount of effort proportional to the square of its size (in its worst case).

References:
Chapter 1
Pages 7-9

Heuristic Algorithms – attempt to find a good solution to a problem, but are not guaranteed to find the optimal or complete solution.

Recursion – defining something in terms of itself .

Examples

Division by repeated subtraction (Pseudo code)

```
Read A,B  (A>B;   A,B>0)
Counter=0
repeat
     Set A to A-B
     Increment Counter
until A<B
Write "Answer" Counter
Write "Remainder" A
```

Draw all connections between n points.
There are n points, so each can be connected to n-1 others. Therefore we could make $n(n-1)$ connections, but we will then have done each one twice. Therefore the total is $\frac{1}{2}n(n-1) = \frac{1}{2}n^2 - \frac{1}{2}n$
When n is large, $\frac{1}{2}n^2$ is considerably bigger than $\frac{1}{2}n$ so this dominates the expression. Since $\frac{1}{2}n^2$ is directly proportional to n^2, we say that this algorithm is $O(n^2)$

For example – Bin Packing algorithms (see below)

Factorial: $10! = 10 \times 9! = 10 \times 9 \times 8! = \ldots\ldots$

References:
Chapter 1
Pages 15-19

Exercise 1C
Q. 1

References:
Chapter 1
Pages 20-24

Exercise 1D
Q. 1

Exercise 1E
Q. 4

Standard Types of Algorithm

Searching
Finding a particular item in a list

Linear	Every data item is checked
Binary	For ordered data. Repeatedly examines the middle to decide which half contains the target.
Index	Search the index to find the "page", then search the page.

Sorting
Placing all items in a list in the correct order

Exchange	Find the smallest and swap it with the first. Find the next smallest and swap it with the second; etc.
Bubble	On the first pass compare adjacent items (1st with 2nd, 2nd with 3rd etc.), and swap if necessary. Repeat until all items sorted.
Quick	Choose a pivot value and sort items into two sub-lists, containing items smaller and larger than the pivot respectively. Repeat algorithm on each sub-list until all sub-lists are of length 1 or 0.

Bin Packing
The process of allocating items to locations, given a set of items of specified sizes and a set of locations with specified capacities.

First fit — Take each item in turn, placing it in the first available slot.
First Fit Decreasing – order the items in decreasing size then apply First Fit.

Mathematical
Leading to the calculation of a required value

HCF, LCM, Factorial

NOTE: Syllabus requires knowledge of properties of algorithms in all topics.

Summary D1 Topic 2: **Graphs**

References:
Chapter 2
Pages 43-45

References:
Chapter 2
Pages 47-50

Exercise 2A
Q. 4

Terminology

Graph – collection of *vertices* & *edges*

Sub graph – any set of edges & vertices taken from a graph is a sub-graph

Vertex/Node – the dots in a graph (usually where 2 or more edges meet, but not necessarily)

Edge/arc – a line between two vertices

Isomorphic - two graphs are isomorphic if one can twisted into the shape of the other ie: by relabelling the vertices of one graph you can show that it is the same as the other

Weight – the weight of an edge is a number representing a real-world value, often distance or time

Multiple edges – two or more edges joining the same pair of vertices

Loop – an edge starting and finishing at the same vertex

Simple graph –a graph with no loops or multiple edges

Connected graph – a graph in which a route can be found between any pair of vertices (ie the graph is in one part)

Cycle – a route starting and finishing at the same vertex

Tree – a graph with no cycles

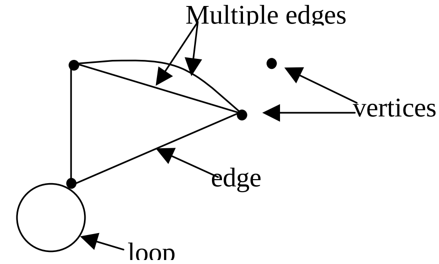
Degree of a vertex – the number of edges starting or finishing at that vertex

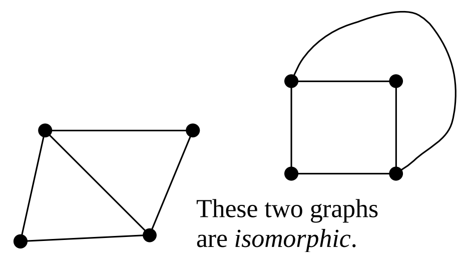
Handshake lemma – the sum of all the degrees in a graph is twice the number of edges (and therefore even); consequently there will always be an even number of odd vertices

Di-graph – a graph in which the edges indicate direction

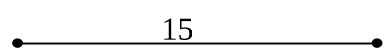
Incidence matrix – a matrix representing the edges in a graph

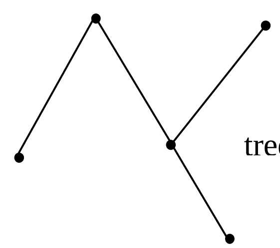
Examples



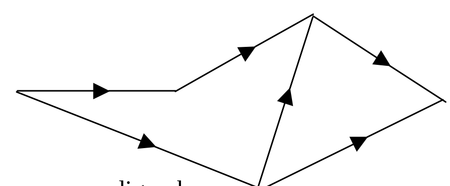


These two graphs are *isomorphic*.



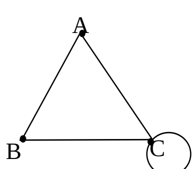


tree



digraph

		To		
		A	*B*	*C*
From	*A*	–	1	1
	B	1	–	1
	C	1	1	2



Incidence matrix and graph

The topic of graphs is fundamental to much of Discrete Maths, and the theory of graphs is a large topic in its own right. However, this course focuses on ways of using graphs. Exam questions could test you on your knowledge of graphs; they could also ask you to apply a graph to solve a practical problem.
Questions like this in Ex 2A, include:2,3,4,5,9,12,15,16,17
Other questions, including some taken from past examinations of other syllabuses, involve terminology and notation not specified in this syllabus. Tackling these, and investigating the terms involved, will enhance your knowledge of this subject, and your modelling skills.
These questions include:6,7,8,10,11,13,14

Summary D1 Topic 3: Networks

References:
Chapter 3
Pages 60-61

Terminology

Network - a graph with weighted arcs (or edges)

References:
Chapter 3
Pages 62-65

Minimum connector – the smallest possible tree which leaves a graph connected (*spanning tree*)

Kruskal's algorithm

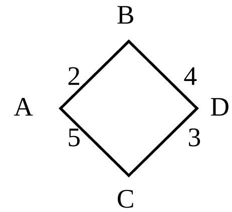
1. Select the shortest edge in a network
2. Select the next shortest edge which does not create a cycle
3. Repeat step 2 until all vertices have been connected

Prim's algorithm

1. Select any vertex
2. Select the shortest edge connected to that vertex
3. Select the shortest edge which connects a previously chosen vertex to a new vertex
4. Repeat step 3 until all vertices have been connected

Exercise 3A
Q. 1

Example



Kruskal's		Prim's (starting from A)	
1	AB	1	AB
2	CD	2	BD
3	BD	3	DC

References:
Chapter 3
Pages 68-79

Shortest path – the shortest route between any two nodes in a network

Dijkstra's algorithm

Step 1 Begin by giving the start vertex permanent label 0, and order label 1.

Put in temporary values for each vertex you can reach directly from the start.

Step 2 Pick the vertex with the smallest temporary value, and make that the permanent value. Put in the correct order label. Then update the temporary values for all the vertices you can reach directly from the vertex you've just made permanent.

Repeat step 2 (always looking at all the temporary values for the smallest), until you have a permanent value at your target vertex.

NB If at any stage you can choose between two equally good options, pick either; you will always pick the other one next.

Step 3 Trace back to find the route - but write it down forwards, giving the total distance covered.

NB If you find yourself with a choice of arcs when tracing back, your network will have more than one optimum solution.

Exercise 3B
Q. 1

Exercise 3C
Q. 2, 3

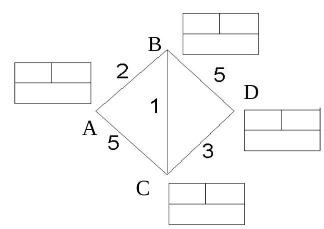
Exercise 3D
Q. 4

References:
Chapter 3
Pages 86-87

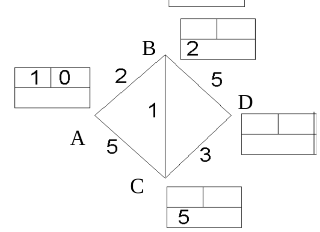
Exercise 3E
Q. 4

Example using Dijkstra, from A –D

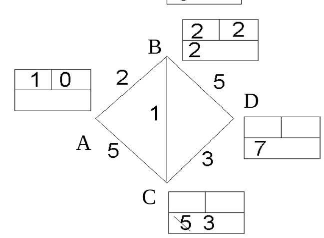
1



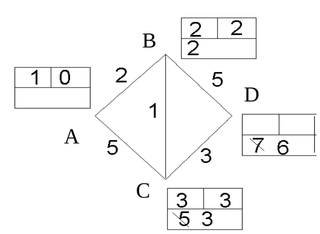
2



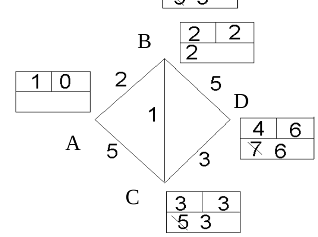
3



4



5



References:
Chapter 4
Pages 101-107

Exercise 4A
Q. 4

Terminology

A Critical Path Analysis question will consist of a project to be completed by carrying out a number of tasks.

Activity – a task to be done which takes a defined amount of time. Represented by an edge on a graph.

Event – an instant in time when one or more activities start or finish. Represented by a vertex.

Precedence Table – lists each activity and those activities which must immediately precede it.

Critical Path – the set of edges (activities) which must be carried out at a fixed time for the project to be completed in the minimum time.

Event times – each event has an earliest and a latest time: these are the earliest that all incoming activities could finish, and the latest that all outgoing activities could commence, without affecting the minimum completion time.

Float - the amount of time by which an activity (not on the critical path) can be delayed or extended without affecting the critical path timing.

- independent float does not affect other activities
- interfering float means that it is dependent on other activities not using all of their float

References:
Chapter 4
Pages 111-115

Exercise 4B
Q. 3

Exercise 4C
Q. 9

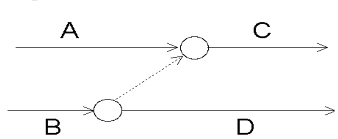
Resourcing
Resourcing is the task of finding how many people are required to complete the project in the minimum time.

1. Each task requires one person. In this case the critical path will usually be completed by a single person. The other tasks are scheduled (possibly making use of float time to delay the start times) so that the least number of people are required [there are links here to the concepts of bin-packing from the algorithms section].

2. Tasks require different numbers of people. A resource histogram should be drawn. Then, as above, start times are adjusted in an attempt to even out the requirments – this is known as *resource levelling*.

Dummy Activity - these are represented by dotted or faint lines and are added for two reasons:
1. To avoid two activities sharing both start and finish events.
2. To ensure correct logic.

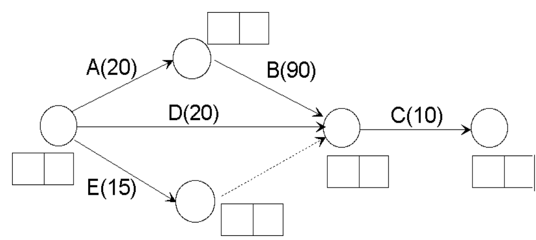
Example: if C is dependent on A and B, but D is only dependent on B



Example
Make a cup of instant black coffee

Activity	Duration (secs)	Preceding activities
A Fill Kettle	20	-
B Boil Kettle	90	A
C Put water in mug	10	B, D, E
D Put coffee in mug	20	-
E Put sugar in mug	15	-

The above precedence table leads to the following graph (activity network)



The events are shown as circles, the activities as edges. All precedences are maintained; a dummy activity is inserted after E so that each activity is uniquely defined by (start event, finish event).

The minimum completion time (assuming tasks can be done in parallel) is 120 seconds. The critical activities for this are A, B and C, since a delay in any of these would lead to a delay in the final completion time.

The event at the end of E has an earliest time of 15 seconds (one cannot get there sooner), and a latest time of 110 seconds (since anything later than this would delay the start of C and hence the entire project).

Activity D has float time of 90 seconds (since it can start at 0, must finish by 110, and takes 20 seconds itself).

References:
Chapter 5
Pages 138-145

Terminology

Variable – a named quantity which can take more than one value.

Objective Function – an expression which is to be maximised or minimised, expressed as a linear function of the variables.

Profit/Cost – the value of the objective function, depending on whether it is a maximisation or minimisation problem.

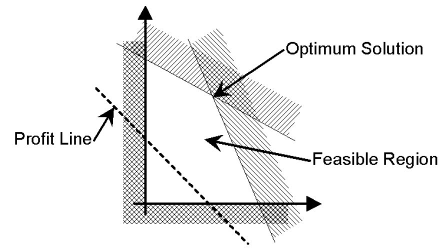
Constraint – an equation or inequality involving at least one variable which limits the value(s) the variables can take.

Feasible region – that part of the graph which represents solutions which satisfy all of the constraints

Optimum solution – the solution offering the best value of the objective function

Examples
x, y, Xab, Product1
maximise $2x + 3y$

$P = 2x + 3y$
$x + 2y \le 12$
$x \ge 3$
$2x - y = 4$



References:
Chapter 5
Pages 141-142

Graphical Solutions
(assuming a two-variable maximisation problem)

Label axes for each *variable.*
For each *constraint*, draw the line representing the equality. For inequalities, shade out the unwanted side (above for <, below for >).
Shade out left of vertical and below horizontal axes.

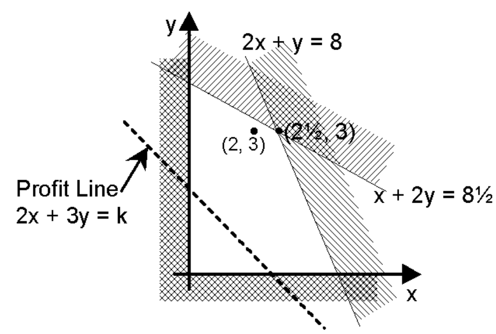
Example 5.1
Page 141-145

The optimum solution will be at one (or two adjacent) vertices of the feasible region. It can thus be found by evaluating the Profit function at each vertex.

Exercise 5A
Q. 2

This point will correspond to the point where a line equal in gradient to the profit function and moving across the feasible region away from the origin, will last touch the feasible region.

Example
Maximise $2x + 3y$ subject to
$2x + y \le 8$, $x + 2y \le 8½$
The gradient of $P = 2x + 3y$ is $-^2/_3$



The optimum solution occurs at the intersection of the lines $2x + y \le 8$,
$x + 2y \le 8½$
which is $x = 2½$, $y = 3$ with a profit of 14.
This can be estimated from the graph, or calculated by solving the simultaneous equations.

References:
Chapter 5
Pages 148-156

Integer problems
If the *optimum solution* is not integer, evaluate the *objective function* at the integer point(s) [usually near to the optimum point] and select the best.

Minimisation problems
These are done the same way at this level.

Multiple solutions
These occur when the *objective function* is parallel to a *constraint*, and occasionally in integer problems.

Exercise 5B
Q. 3

Exercise 5C
Q. 3

Spare capacity
Any constraint which does not pass through the optimal solution will have spare capacity which can be calculated. This can be done by substituting the values for the variables and noting the difference across the inequality sign.

If Integer solutions are required, then the nearest integer point in the feasible region is (2, 3) so the solution is $x = 2$, $y = 3$ with a profit of 13. However, it is important to check other integer points nearby (e.g. 2, 4) to see if they are also feasible and give a greater profit.

Multiple solutions would occur if the profit function changed to say $P = 2x + y$: it would be all points on the constraint line between (2½, 3) and (4,0).

At the integer solution the constraints have spare capacity of 1 and $^1/_2$ respectively.

References:
Chapter 6
Pages 168-169

Exercise 6A
Q. 3

References:
Chapter 6
Pages 173-174

Exercise 6B
Q. 1

Terminology

Simulation: a mathematical model that can be used to test what might happen in situations where an experiment with real subjects may be too dangerous or take too long.

Deterministic model: assumes that chance events do not occur.

Stochastic model: allows for an element of chance.

Random variables: used in stochastic models to simulate probabilities. Of two types

Uniformly distributed random variables

Non-uniform random variables

Rules of the simulation and Cumulative frequency methods: allocating the random variables in line with the probabilities

Inter arrival times: how often a new customer joins the queue

Examples

What system of queuing will minimise customer waiting times in a bank?

How does an infectious disease spread through a population?

Queuing simulations are a common example

coins or dice

random number tables, random number generated from a calculator

including rejecting values where necessary

References:
Chapter 6
Page 174

Exercise 6C
Q. 3

References:
Chapter 6
Page 178

Exercise 6D
Q. 4

Queuing simulations

1. Decide on the important variables

2. Make simplifying assumptions

3. Write the rules of the simulation.
Use the probabilities to allocate random numbers. These are usually 2 digit numbers from 00 to 99.

100/6 = 16 to nearest whole number. Allocate random numbers in multiples of 16 and reject any which are left over.

4. Perform the simulation
Use the random numbers to obtain the inter-arrival times, the actual arrival times and service times.
Inter arrival time random numbers: 13, 24, 88, 12

Service time random numbers: 58, 73, 11, 98, 39
(Ignore 98)

In the table, the beginning of the simulation is usually counted as time 0 and subsequent arrival times are given as time from the beginning of the simulation.

5. Interpret the results
How does this compare with reality?

6. Amend the model
Often the first run is to validate the model – to check that it models accurately the current or known situation.

When modelling queuing it is necessary to find out the distributions for inter-arrival times and serving time.

All times are taken to the nearest minute
Any times which fall outside this range will be ignored.

Inter-arrival time (mins)	1	2	3
probability	0.2	0.5	0.3
Random Numbers	00-19	20-69	70-99

Service time (mins)	1	2	3
probability	$\frac{1}{6}$	$\frac{1}{2}$	$\frac{1}{3}$
Random Numbers	00-15	16-63	64-95

The simulation is usually set out in a table.

customer	RND	Arrival time	Service start	RND	Service time	Service ends	Queing time
1			0	58	2	2	0
2	13	1	2	73	3	5	1
3	24	3	5	11	1	6	2
4	88	6	6	39	2	8	0

$$\text{Average queuing time} = \frac{\text{total queuing time}}{\text{number of customers}} = \frac{3}{4} \text{ min per customer}$$

Note: the above simulation starts at the arrival of the first customer, otherwise an initial arrival time would be needed.
A simulation would normally have a much longer run than this.
A simulation should be performed several times to get a more reliable result.