OCR Computer Science A Level

2.2.2 Computational Methods

Intermediate Notes

Specification:

2.2.2 a)

- **Features that make a problem solvable by computational methods**

2.2.2 b)

- **Problem recognition**

2.2.2 c)

- **Problem decomposition**

2.2.2 d)

- **Use of divide and conquer**

2.2.2 e)

- **Use of abstraction**

2.2.2 f)

- **Solving problems using:**
    - Backtracking
    - Data mining
    - Heuristics
    - Performance modelling
    - Pipelining
    - Visualisation

Features that make a problem solvable by computational methods

The first stage of problem solving is identifying whether or not a problem can be solved using computational methods. A problem that can be solved using an algorithm is computable. Problems are computable only if they can be solved within a finite, realistic amount of time. Problems that can be solved computationally typically consist of inputs, outputs and calculations.

Although some problems are computable, it may be impractical to solve them due to the resources (processing power, speed and memory) or time they require in order to be completed.

Problem recognition

Stakeholders state what they require from the solution and this information is used to clearly define the problem and system requirements. Requirements may be defined by:

- Analysing strengths and weaknesses with the current solution
- Considering inputs, outputs, stored data and volume of data

Problem decomposition

Once a problem has been defined, it is broken down into smaller problems until each subproblem can be represented as a self-contained subroutine. This technique is called problem decomposition and aims to reduce the complexity of the problem by splitting it up into smaller sections. By identifying subproblems, programmers may find that certain sections of the program can be implemented using pre-coded modules or libraries.

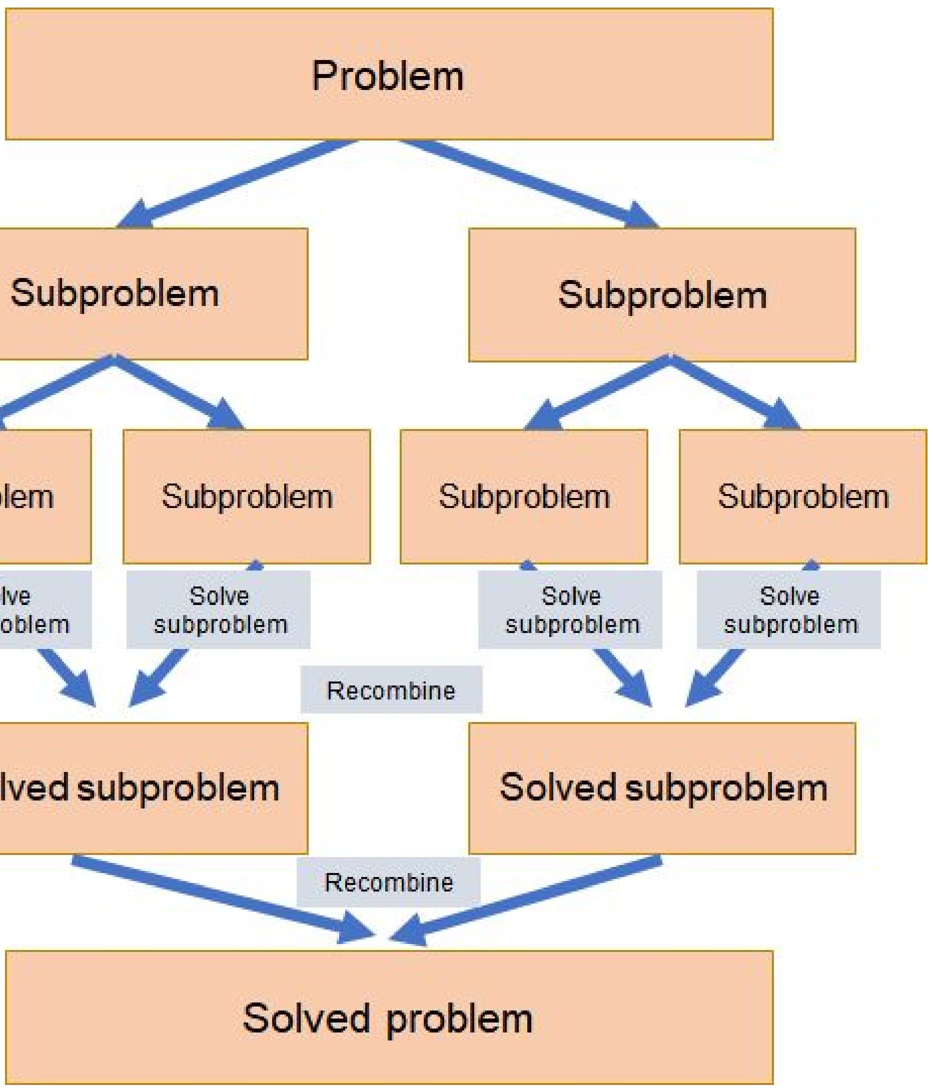
Synoptic Link

Top-down design discussed in **2.2.1** is an example of **problem decomposition**.

Decomposition also makes the project easier to manage, as different teams can be assigned separate sections of the code. These can be individually designed, developed and tested before being combined at the end. This makes it possible to develop modules in parallel, therefore deliver projects faster. This technique also makes debugging simpler and less-time consuming, as it is easier to identify, locate and mitigate errors.

Use of divide and conquer

Divide and conquer is a problem-solving technique that can be broken down into three parts: divide, conquer and merge. 'Divide' involves halving the size of the problem with every iteration. Each subproblem is solved in the 'Conquer' stage, often recursively. The solutions to the subproblems are then recombined during the 'Merge' stage to form the final solution to the problem.



Divide and conquer is applied to problem-solving in quick sort, merge sort and binary search. The biggest advantage of using divide and conquer to solve problems is that the size of the problem is halved with each iteration which greatly simplifies very complex problems. This means that as the size of a problem grows, the time taken to solve it will not grow as significantly. As divide and conquer mostly makes use of recursion, it faces the same problems that all recursive functions face: stack overflow will cause the program to crash and large programs are very difficult to trace.

Use of abstraction

Representational abstraction is a powerful technique that is key to solving problems computationally. This is when excessive details are removed to simplify a problem. Problems may be reduced down until they form problems that have already been solved which allows pre-programmed modules and libraries to be used rather than coding from scratch.

> **Synoptic Link**
>
> You will have already encountered **abstraction** in **2.1.1**.

Using levels of abstraction allows for the functionality of a large, complex project to be split up into simpler component parts which can then be dealt with by different teams, with details about other layers being hidden. This technique makes projects more manageable.

Abstraction by generalisation may also be used to group together different sections of the problem with similar functionality. This means segments can be coded together, saving time. Abstraction is also required to represent real-world entities with computational elements, such as tables and variables.

Problem solving strategies

There are several other techniques that may be used to solve problems computationally:

Backtracking

Backtracking is a problem-solving technique implemented using algorithms, often recursively. It works by methodically visiting each path and building a solution based on the paths found to be correct. If a path is found to be invalid, the algorithm backtracks to the previous stage and visits an alternate path. Depth-first graph traversals are an example of backtracking.

> **Synoptic Link**
>
> You will learn more about **graph traversals** such as **depth-first** in **2.3.1**.

This process can be visualised using the idea of a maze. A maze is solved by visiting each path and, if a path leads to a dead end, returning back to the most recent stage where there are a selection of paths to choose from.

Data mining

Data mining is a technique used to identify patterns or outliers in large sets of data collected from a variety of sources, termed big data. Data mining is used in software designed to spot trends or identify correlations between data which are not immediately obvious.

Insights from data mining can be used to make predictions about the future based on previous trends. This makes data mining a useful tool in assisting business and marketing decisions. Data mining has also been used to reveal insights about people's shopping habits and preferences based on their personal information. These insights can be used to inform marketing techniques.

As data mining often involves the handling of personal data, it is crucial that it is dealt with in accordance with the present legislation regarding data protection. As of 2018, all data held and processed by organisations within the EU must follow the rules set by GDPR.

Synoptic Link

You will have learnt about laws surrounding **data protection** detail **1.5.1**.

Heuristics

A heuristic is a non-optimal, 'rule-of-thumb' approach to problem-solving which is used to find an approximate solution to a problem when the standard solution is unreasonably time-consuming or resource-intensive to find. The solution found through using heuristics is not perfectly accurate or complete.

Heuristics are used to provide an estimated solution for intractable problems such as the renowned Travelling Salesman Problem as well as the A* algorithm, and are also used in machine learning and language recognition.

Intractable problems

Problems for which the solutions **takes an unreasonably long time** to be found.

Performance modelling

Performance modelling eliminates the need for true performance testing by providing mathematical methods to test a variety of loads on different operating systems. This provides a cheaper, less time-consuming or safer method of testing applications. It is useful for safety-critical computer systems, where it is not safe to do a real trial run. The results of performance modelling can help companies judge the capabilities of a system, how it will cope in different environments and assess whether it is safe to implement.

Pipelining

Pipelining is a process that allows for projects to be delivered faster, as modules are divided into individual tasks, with different tasks being developed in parallel. Traditionally, the output of one process in pipelining becomes the input of another, resembling a production line. Pipelining is commonly used in RISC processors, in which the different sections of the Fetch-Decode-Execute cycle are performed simultaneously.

Visualisation

Data can be presented in a way that is easier for us to understand using visualisation to produce graphs, trees, charts etc. This makes it possible to identify trends that were not otherwise obvious. Visualisation is another technique that is used by businesses to identify patterns which can be used to inform business decisions.