

Unit 6.2: We are computational thinkers



Mastering algorithms for searching, sorting and maths

Software: Google Maps, Scratch (alternative: Snap!)

Hardware: Laptop/desktop/Chromebook computers or iPads, unplugged resources

Overview

In this unit, pupils participate in some hands-on unplugged activities which help them to develop an understanding of some important **algorithms**. They also investigate these when implemented as Scratch or Snap! programs. In:

- **Session 1** they find the shortest route between towns
- **Session 2** they find the smallest number of coins needed to make change
- **Session 3** they learn about random and **linear search** algorithms

- **Session 4** they learn about **binary search** algorithms
- **Session 5** they learn about **selection sort** algorithms
- **Session 6** they learn about **quicksort** algorithms.

Alternatives

Pupils could use Snap! as an alternative to Scratch for the programming activities here. Bing or Apple Maps can be used as an alternative to Google Maps in Session 1.

Knowledge, skills and concepts

In this unit, pupils will learn to:

- develop the ability to reason logically about **algorithms**
- understand how some key algorithms can be expressed as programs
- understand that some algorithms are more efficient than others for the same problem
- understand common algorithms for **searching** and **sorting** a list.

Progression

In Key Stage 1:

- Pupils thought about recipes as sequences of instructions in **Unit 1.2: We are TV chefs**.

- Pupils thought about the sets of rules for some simple computer games in **Unit 2.2: We are game testers**.

In Key Stage 2:

- Pupils used logical reasoning to detect and correct errors in programs in **Unit 3.2: We are bug fixers**.
- Pupils were introduced to the idea of a **graph** linking locations in an interactive adventure game in **Unit 5.5: We are adventure gamers**.
- Pupils are introduced to some of the algorithms for machine learning and other aspects of artificial intelligence in **Unit 6.6: We are AI developers**.

Assessment – by the end of the unit:

All pupils can:

- use Google Maps to find the shortest or fastest route between two places
- work out the smallest number of coins needed to make an amount of change
- use random, linear and **binary search** to play the 'Guess my number' game
- sort yoghurt pots into order with a balance pan, using their own **algorithm** and **quicksort**.

Most pupils can:

- find optimum routes on a simplified map
- record an algorithm for finding the smallest number of coins to make change

- record algorithms for random, linear and **binary search**
- record an algorithm for sorting
- appreciate that quicksort will be faster than, e.g. **selection sort**.

Some pupils can:

- find the shortest set of roads to connect towns
- create a Scratch program to work out the smallest number of coins needed to make change
- correct Scratch and Snap! programs which implement search and sort algorithms.

Background information

- Pupils have been learning about the computational thinking concept of **algorithms** (a sequence of steps or set of rules for solving a problem) from Year 1 onwards. In this unit, they consider a more abstract set of problems – from finding routes and making change, to **searching** and **sorting** programs.
- Most of these sessions start with ‘unplugged’ activities, where pupils think through the algorithms away from the computer. They then connect their problem-solving approach with programming, and automating their solutions in Snap! or Scratch.

Key vocabulary

Abstraction: computational thinking approach to managing complexity by simplifying things, through identifying what is important and what detail can be hidden or ignored

Algorithm: a sequence of precise instructions or steps (sometimes a set of rules) to achieve an objective

Binary search: search algorithm that identifies repeatedly which half of the list of possible elements the target belongs to

Decomposition: breaking a problem down into smaller parts

Divide and conquer: class of algorithms in which the problem is decomposed into smaller, simpler parts, to which the same algorithm can then be applied

Graph: data structure showing the connections between elements

Greedy algorithm: algorithms that work on a ‘biggest first’ basis, applying divide and conquer methods to reduce the problem rapidly to a simpler problem

Linear search: search algorithm that looks at each element in turn to see if it meets the criteria

Quicksort: ‘divide and conquer’ sort algorithm which partitions a list into elements smaller than and larger than a pivot element, and then applies the algorithm to sorting each of these lists

Search: to identify an element of a list that meets specified criteria

Search algorithm: the way results for a search are selected and ranked, typically through the presence of key words, and by the number and quality of inbound links

Selection sort: sort algorithm which looks for the largest element, then the next largest and so on, until the list is in order

Sort: to put a list into order

Differentiation

See each session (pages 23–28) for ways to increase support and add challenge to this unit. To challenge, encourage pupils to write the steps or rules of their **algorithms** down, possibly recording these as a diagram. Encourage them to translate their algorithms into programs, testing whether these work as they should. Ask pupils to think about whether one approach to solving a problem is better than another, and what this means.

For pupils who find these ideas difficult, focus on the practical, ‘unplugged’ activities at the beginning of each session, making use of physical manipulative, practical equipment. For the programming activities, provide pupils with partially completed programs or Scratch ‘jigsaw’ puzzles, rather than expecting them to work from a blank screen.

Cross-curricular opportunities

- **Geography:** Pupils can learn about local routes, transport and destinations.
- **Maths:** Pupils can investigate how many ways to make change in money, play ‘Guess the number’ game, develop mathematical fluency by asking different types of questions and ordering numbers, decimals, percentages and fractions.
- **English:** Pupils can search for words in a dictionary and arrange lists into alphabetical order.

Preparation for teaching the unit

Things to do

- Decide which software/tools are most accessible and appropriate for use with your class.
- Watch the walkthrough videos.
- Watch the CPD videos (see *Additional resources*).
- Spend some time familiarising yourself with your chosen software/tools.
- Ensure you have devices booked in advance.
- Prepare any unplugged resources that you are going to use: for the sort activities, you will need at least one set of eight opaque yoghurt pots or Smarties tubes, each containing a different mass (coins would work well) and a pan balance.
- Decide how you will organise the class for the unit – pairs would work well. The pan balance activities can be done as a whole class.
- Work through the unit yourself.

Resources needed

- **Software:** Google Maps, Scratch, Snap!
- **Hardware:** Laptop/desktop/Chromebook computers or iPads, unplugged resources

Online resources provided

Session resources

- Worksheet 6.2a: Network graph
- Worksheet 6.2b: Guided question sheet
- Worksheet 6.2c: End-of-unit quiz
- Worksheet 6.2d: Pupil self-assessment
- Teaching slides 6.2a–6.2f
- Walkthrough videos 6.2a–6.2g
- Interactive end-of-unit quiz 6.2

Additional resources

- CPD video: Comparing search algorithms
- CPD video: Comparing sort algorithms

Alternatives

- Software in 60 seconds: Introduction to Snap!
- Software in 60 seconds: Scratch 1–7

Online safety

- Pupils can use Scratch without accounts, but need a parent/carer email address to join the online community.
- While projects and comments on the Scratch community can be deleted by moderators if necessary, there is no automatic moderation. It would be advisable to remind pupils of what is appropriate and inappropriate behaviour, as well as to whom they should report concerns about content.

- Snap! can be used without registering an account. There is no age restriction, but it would be wise to seek a parent or carer's consent.
- When using Google Maps or other online services, make sure that the necessary Internet filtering and logging precautions are in place.

Collaboration

In this unit, pupils discuss how to solve logical problems, and how different tools work – such as search. They also discuss the nuances between different types of search and sort. They work together to solve problems, and evaluate and assess different search types. Pupils collaborate on computational thinking activities, analyse their thinking and determine how this can be applied to problem solving. The programming problems here are best tackled by pupils working together.

Useful links

Software and tools

- Scratch online at: scratch.mit.edu or download from: scratch.mit.edu/download
- Snap! online at: snap.berkeley.edu or download the zip archive from: www.github.com/jmoenig/Snap
- Google Maps: www.google.co.uk/maps
- Scratch programs and solutions for random search jigsaw, linear search jigsaw, binary search jigsaw and selection sort: scratch.mit.edu/studios/27311495/
- Snap! quicksort buggy program: snap.berkeley.edu/snap/snap.html#present:Username=mgberry&ProjectName=buggyquicksort
- Snap! quicksort solution: snap.berkeley.edu/snap/snap.html#present:Username=mgberry&ProjectName=quicksort


Online tutorials

- Scratch and Snap! tutorials in editors

Information and ideas

- Examples of search and sort activities from CS Unplugged: www.csunplugged.org/activities
- BBC Bitesize search: www.bbc.co.uk/education/guides/zgr2mp3/revision
- BBC Bitesize sort: www.bbc.co.uk/education/guides/z2m3b9q/revision
- QuickStart Computing on computational thinking: <https://community.computingatschool.org.uk/files/8221/original.pdf>
- Dijkstra's shortest path algorithm: www.youtube.com/watch?v=GazC3A4OQTE

Session 1: Finding routes

Learning objective	To find the shortest routes on a map.
<p>Steps and activities</p>  <ul style="list-style-type: none"> ● TS6.2a ● V6.2a ● WS6.2a 	<p>In this session, pupils look for possible routes on a map, finding the shortest.</p> <ol style="list-style-type: none"> 1 Ask pupils to think of the route from their school to a familiar location, such as a shop in the local town centre. 2 Ask pupils to write down the sequence of instructions for how to get from the school to the location. Discuss the following questions: <ul style="list-style-type: none"> ● Has everyone chosen the same route? ● If not, how could you decide which route is best? ● Explain that the sequence of instructions is an algorithm. What other algorithms have pupils written in computing lessons? 3 Show pupils Google Maps and ask it to find a route from the school to the chosen location (you could use TS6.2a/V6.2a to model this). <ul style="list-style-type: none"> ● Show the sequence of driving instructions. ● Show how it can give walking, cycling and public transport routes. ● Explain that these routes are also algorithms – for these, the computer has followed one algorithm to then work out other algorithms. 4 If you have road atlases or maps available, share these with pupils and ask them to work out a route from one town to another. Discuss the following questions: <ul style="list-style-type: none"> ● How do they know that it is the shortest (or fastest) route? ● Ask them to try the same journey on Google Maps. ● Did Google Maps find the same route? 5 Show pupils the simplified map of actual towns and roads – called a network graph – on TS6.2a and WS6.2a. On here, the important information has been kept but other details have been hidden away. Ask them to find the shortest route they can from U to G on the map. How do they know this is the shortest route? 6 Give pupils some other pairs of places (nodes) on the simplified map, again asking them to find the shortest routes between these places. Ask pupils to describe their algorithm for finding shortest routes. See an example on TS6.2a. 7 Explain that one algorithm could be finding all of the possible routes, working out the total distance for each and picking the shortest. <ul style="list-style-type: none"> ● Explain that this is a slow method and that computer scientists want to find quicker algorithms for solving a problem, just as one would want to find the quickest route when travelling. ● Google Maps uses very efficient algorithms to find the shortest or fastest routes – these involve breaking down journeys into smaller parts and finding the best routes for these smaller journeys. This approach is called 'divide and conquer' and is based on the idea of decomposition.
<p>Challenge</p>	<p>Pupils could:</p> <ul style="list-style-type: none"> ● solve other problems on the graph provided, such as finding the shortest set of roads that connect all of the towns (a minimum spanning tree), or the shortest route that visits all of the towns (the 'travelling salesman' problem) ● investigate some of the standard algorithms for solving this problem, such as Dijkstra's shortest path algorithm (see Useful links page 22).
<p>Support</p>	<p>Pupils could:</p> <ul style="list-style-type: none"> ● solve the problem on a simpler graph with smaller numbers ● experiment with finding routes between familiar places using Google Maps.
<p>Homework</p>	<ul style="list-style-type: none"> ● Explore Google Map's routes with parents or carers for familiar and unfamiliar journeys. ● Discuss how satnav directions are worked out with their parents or carers.

Key to online resources


WS = Worksheet

TS = Teaching slides




Q = Quiz

V = Video


Session 2: Finding the smallest number of coins

Learning objective	To find the smallest number of coins to make change.
<p>Steps and activities</p>  <ul style="list-style-type: none"> ● TS6.2b ● V6.2b 	<p>In this session, pupils record an algorithm for finding the smallest number of coins to make a given amount of change. They write a Scratch program to implement their algorithm.</p> <ol style="list-style-type: none"> 1 Ask pupils what different ways there are to make, e.g. 8p, using normal British coins. Which way uses the smallest number of coins? (5p, 2p and 1p.) 2 Explain that vending machines must have some way of working out the best way to give change. The best way is the one that uses the smallest number of coins. 3 Show pupils the amounts on TS6.2b and ask them to work out the smallest number of coins needed to give each of these amounts as change. 4 Can pupils write down the steps that they go through to find the smallest number of coins for an amount of change? <ul style="list-style-type: none"> ● Their algorithm is likely to be a 'greedy algorithm', which starts with the largest possible coin value and reducing the problem to progressively smaller amounts as it goes. ● Think how 28p would be given in change. They would give 20p as the largest possible coin first. Then they would have to give change for the remaining 8p, so they would give 5p, then 2p, then 1p. ● Explain that this algorithm also uses decomposition (as 'divide and conquer'), in each case reducing the problem to a simpler one and applying the same algorithm to that. 5 Ask pupils if they can think of how they might code their algorithm in Scratch. <ul style="list-style-type: none"> ● Have them share ideas together as a class. ● Explain that their program will need to keep track of the running total and the number of coins given so far. They will need to use variables for this. ● Pupils will also need to test if the running total is more than the coin value. For example, if the running total is 15p and the coin value is 20p, the algorithm needs to be able to recognise this. ● If the running total is more, they can remove the coin value from the total and add one to the number of coins. For example, if pupils have used two coins, and the running total is 15p and the coin value is 10p, pupils take 10p away from 15p (leaving 5p) and add one to the number of coins – making this three coins. Otherwise, they can move on to the next smaller coin. 6 Give pupils some time to work on their program with a partner (you could use TS6.2b/ V6.2b to model steps 6 and 7). <ul style="list-style-type: none"> ● Ask them to test their program thoroughly to make sure it works as it should. ● Ask them to check that it gives the correct answer for all the amounts on TS6.2b. 7 Can pupils think of ways to improve their program? <ul style="list-style-type: none"> ● Could they use Scratch's 'make a block' tool to simplify their program? ● Ask pupils to think about solving the problem with different currency systems. ● What if the coin values were £1.28, 64p, 32p, 16p, 8p, 4p, 2p and 1p? ● How would they change their program to work with these coin values? ● Explain that these are important numbers in computing: computers represent all numbers using either 0 or 1 of these values: this is called the binary number system.
<p>Challenge</p>	<p>Pupils could:</p> <ul style="list-style-type: none"> ● simplify their program, using the 'make a block' tool and lists in Scratch ● convert their program to work out the binary values for decimal numbers using the same 'greedy algorithm'.
<p>Support</p>	<p>Pupils could:</p> <ul style="list-style-type: none"> ● be given an algorithm for change making to convert to a Scratch program ● be provided with a partially completed program, or just the blocks needed to create their Scratch program ● find it helpful to have plastic or real coins available for the activity.
<p>Homework</p>	<p>Ask pupils to think of the smallest number of normal or binary coins for amounts they see when visiting shops.</p>


Session 3: Random and linear search

Learning objective	To understand random and linear search algorithms.
<p>Steps and activities</p>  <ul style="list-style-type: none"> ● TS6.2c 	<p>In this session, pupils play maths games that represent random and linear searches, before thinking about how it might be applied as an algorithm in Scratch.</p> <ol style="list-style-type: none"> 1 Introduce pupils to the ‘Guess my number’ game, which they may have played in maths. <ul style="list-style-type: none"> ● In the game, you think of a number and then pupils take it in turns to ask ‘Yes’ or ‘No’ questions to work out what it is. ● Play the game as a class, choosing a number from 0 to 127 and recording pupils’ questions and your answers on the whiteboard. ● For example, questions could be: ‘Is it 23?’, ‘Is it even?’, ‘Does it end in a 3?’ or ‘Is it less than 50?’. 2 Put pupils into pairs and have them work with a partner to play the game a few times. 3 Ask pupils to think about how they might program a computer to play this game. <ul style="list-style-type: none"> ● Emphasise that they would need to think of an approach that could be expressed as an algorithm. ● Ask pupils to record their ideas, perhaps as a flow chart or as pseudocode (i.e. as instructions in English to represent the steps of their algorithm). 4 Get pupils to swap their algorithms with another pair and test them. 5 Show pupils the random search algorithm on TS6.2c. <ul style="list-style-type: none"> ● Ask them to explain in their own words how this works (the computer guesses randomly in the range 0 to 127 until it guesses correctly). ● Ask pupils whether they have ideas for how this could be improved. 6 Show pupils the linear search algorithm on TS6.2c. Ask if they can explain this in their own words (the computer starts at 0, then increases its guess by one each time until it guesses correctly). Ask pupils to evaluate the algorithm.
<p>Challenge</p>  <ul style="list-style-type: none"> ● V6.2c ● V6.2d 	<p>Show pupils the Scratch linear and random search jigsaws (see <i>Useful links</i> page 22). Ask them to choose and complete one of the programs. Pupils should test the programs and assess how efficient they are. There are solutions provided on V6.2c–d and also links provided to the solutions in <i>Useful links</i> page 22.</p>
<p>Support</p>  <ul style="list-style-type: none"> ● WS6.2b 	<p>Pupils could:</p> <ul style="list-style-type: none"> ● play with numbers 0–20 ● use the guided question sheet (WS6.2b).
<p>Homework</p>	<p>Encourage pupils to think of any examples of real-world search algorithms in the home, such as matching socks in a drawer, books on a shelf or video games on a rack. Could their algorithms in these situations be made more efficient?</p>


Session 4: Binary search

Learning objective	To understand binary search algorithms.
<p>Steps and activities</p>  <ul style="list-style-type: none"> ● TS6.2d ● V6.2e ● V6.2f 	<p>In this session, pupils learn the binary search pattern, use it in a game and apply it as an algorithm in a Scratch program.</p> <ol style="list-style-type: none"> 1 Ask pupils to explain the two search algorithms that they looked at in the last session. What problems did they have? (Both are slow and random search may never guess the right number.) 2 Can pupils come up with a better way to play the game? Explain the binary search (divide and conquer) algorithm, in which the range of possible numbers is cut in half each time – so, starting with 0 to 127, one might ask: ‘Is it bigger than 64?’, if false then: ‘Is it bigger than 32?’, if true then: ‘Is it bigger than 48?’, until the number is reached. See TS6.2d for a diagram. 3 Ensure pupils are confident in using this method to play the game with a partner. Ask if there are any disadvantages to this algorithm or if it could be sped up further still. 4 Ask pupils to think how they could implement this algorithm as a Scratch program. <ul style="list-style-type: none"> ● Show them the binary jigsaw Scratch program (see <i>Useful links</i> page 22) and ask them to work with their partner to complete and thoroughly test this. ● It should only take seven questions for the program to guess the correct number. ● There is a solution modelled on V6.2e and in <i>Useful links</i>, page 22). 5 Ask pupils to predict how many goes it would take Scratch to guess a number between 0 and 1,023 (just 10) or 0 and 1,048,575 (just 20). <ul style="list-style-type: none"> ● Ask them to edit their code to make either of these the original range and to test their program thoroughly. You could use TS6.2d/V6.2f to model this. ● Are pupils surprised by how quickly Scratch guesses their answer? ● How long would the algorithms in the previous step have taken to guess a number in this range? 6 Explain that computer science is often less about making faster computers, and more about finding faster ways to solve the same problem. This ‘divide and conquer’ type of algorithm is a common one in computing and has wide applications.
<p>Challenge</p>	<p>Pupils could:</p> <ul style="list-style-type: none"> ● implement this program in Scratch without the scaffolding of the jigsaw ● improve the user interface of the program.
<p>Support</p>	<p>Pupils could:</p> <ul style="list-style-type: none"> ● play this game with a smaller number range ● play ‘Guess who?’ using the pirate cards from Unit 1.6: We are detectives.
<p>Homework</p>	<p>Encourage pupils to look for where they could use a version of this ‘divide and conquer’ algorithm for search problems in or around the home. Would a version of this work for playing hide and seek? How might this be adapted for playing a game such as ‘Twenty Questions’?</p>

Session 5: Selection sort

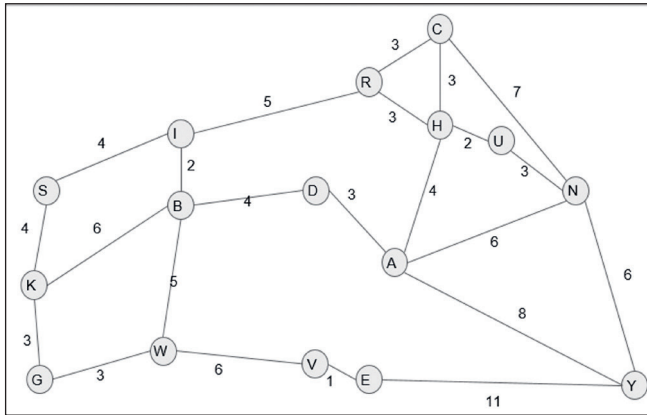
Learning objective	To understand selection sort algorithms.
<p>Steps and activities</p>  <ul style="list-style-type: none"> ● TS6.2e ● V6.2g 	<p>In this session, pupils work in pairs to think of possible algorithms for a sort puzzle. They then test their algorithms with other pairs before fixing bugs in a Scratch selection sort program.</p> <ol style="list-style-type: none"> 1 Show pupils the sort puzzle equipment: a set of eight yoghurt pots/Smarties tubes with different weights inside and a pan balance. The challenge is to sort these into weight order. 2 Pupils should work in pairs to think through a possible algorithm for this challenge. Ask them to record their algorithm, perhaps as a flow chart or as pseudocode. 3 Encourage pupils to test their algorithms out, perhaps using paper slips with different numbers on to represent the pots. 4 Ask pupils to swap their algorithms with another pair. They should try to explain the algorithm in their own words and then try to work out if there are any mistakes in it, ideally by logical reasoning, but perhaps by trying it out with numbered slips. 5 Show pupils the selection sort algorithm (TS6.2e) – in this algorithm, the computer searches through for the heaviest mass, then the next heaviest, then the next heaviest, and so on, until all of the masses are sorted. Ask some pupils to demonstrate this using the pan balance; others could work through this with their partner using numbered slips. 6 Ask pupils to complete the buggy Scratch program for selection sort (see <i>Useful links</i>, page 22) and to test this out with some data. Provide time for pupils to fix the bugs in this code. (The place = length block and looking at = length blocks should be place > length and looking at > length blocks respectively). Less confident pupils might be happier experimenting with the solution (see <i>Useful links</i> page 22 or use the models provided on TS6.2e/V6.2g). 7 Discuss the following questions with pupils: <ul style="list-style-type: none"> ● What do you think about this algorithm? ● Will it always work? ● Can you think of any way to make it faster? ● Would it work as well for a very long list of numbers? 8 Encourage pupils to think of applications for this algorithm around the school, such as: <ul style="list-style-type: none"> ● putting a list of marks into order ● sorting out library books into alphabetical order ● organising a group of children in height order for a school photograph.
<p>Challenge</p>	<p>Pupils could:</p> <ul style="list-style-type: none"> ● implement this program in Scratch rather than debugging the code that is provided ● implement this algorithm as a Python program.
<p>Support</p>	<p>Pupils could:</p> <ul style="list-style-type: none"> ● do the session physically with objects ● complete this challenge with a series of numbers ● work with an adult.
<p>Homework</p>	<p>Ask pupils to look at opportunities to practise sorting at home, e.g. crockery into size order, clothing into an order by colour (e.g. red to violet) or video games into alphabetical order by title. Does this algorithm seem an efficient way to do this?</p>

Session 6: Quicksort

Learning objective	To understand quicksort algorithms.
<p>Steps and activities</p>  <ul style="list-style-type: none"> ● TS6.2f ● WS6.2c ● WS6.2d ● Q6.2 	<p>In this session, pupils find a faster method for sorting numbers – called quicksort. They practise the algorithm without computers first, before applying with computers in pairs.</p> <ol style="list-style-type: none"> 1 Ask pupils to recall what they can about the selection sort algorithm from the previous session. Ask one or more pupils to demonstrate this using the yoghurt pots/Smarties tubes and the pan balance. 2 Ask if any pupils found a faster method for sorting numbers. <ul style="list-style-type: none"> ● Show the class the algorithm for quicksort on TS6.2f, in which all masses are compared with one of the group, thereby splitting the set into those that are lighter and those that are heavier. ● The lighter weights are sorted using the same method, picking one of these to compare with in order to make two subsets – some lighter and some heavier, etc. 3 Encourage pupils to practise this method, using the pan balance or slips of paper. If time allows, they could practise this algorithm in other contexts, such as alphabetical order for a group of children or library books by author. 4 Did pupils notice that this was quicker than selection sort? <ul style="list-style-type: none"> ● Ask pupils to work in pairs, with one counting the number of steps as the other works through the algorithm. ● With two computers, pupils could set up a race using one algorithm against the other or using built-in timers to see which completes the sort more quickly. (They could use the Scratch selection sort and Snap! quicksort programs for this – see <i>Useful links</i> page 22.) 5 Encourage pupils to think of when they might use computers to sort lists into an order, for example: <ul style="list-style-type: none"> ● iTunes libraries or Spotify by artist or title ● library catalogues or search results by author ● school class lists by surname or date of birth ● search results by relevance. <p>The significant time saving made by using quicksort has made these processes much faster than using an algorithm such as selection sort.</p> 6 Finish the unit in the following ways: <ul style="list-style-type: none"> ● Carry out the end-of-unit quiz as a whole class to check understanding. This can be displayed on the whiteboard and done interactively or handed out as a worksheet (Q6.2/TS6.2f/WS6.2c). ● Hand out pupil self-assessment sheet (WS6.2d). Read through the statements with pupils and ask them to fill out the sheet.
<p>Challenge</p>	<p>Pupils could use the buggy Snap! program (see <i>Useful links</i> page 22):</p> <ul style="list-style-type: none"> ● Minimise the stage and draw their attention to the quicksort and join function blocks on the variables palette, which were created for this program. ● Show pupils how they can look inside and edit these blocks. ● Encourage them to experiment with these blocks to test the program. ● Does it always sort the list into order? ● Can pupils fix the program so that it works? (Either < head or > head blocks need to be replaced with < head or = head or > head or = head blocks).
<p>Support</p>	<p>Pupils could:</p> <ul style="list-style-type: none"> ● do the activity with physical objects ● complete this challenge with a series of numbers ● work with an adult.
<p>Homework</p>	<p>Pupils could use a quicksort algorithm for the domestic examples of sort tasks suggested in the previous session. Have pupils noticed that this algorithm is more efficient?</p>

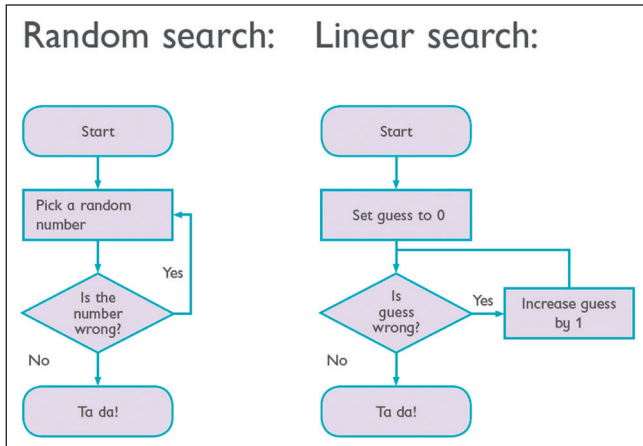
Unit outcomes

Below are some examples of the outcomes you could expect from this unit.



Session 1: Using a network graph

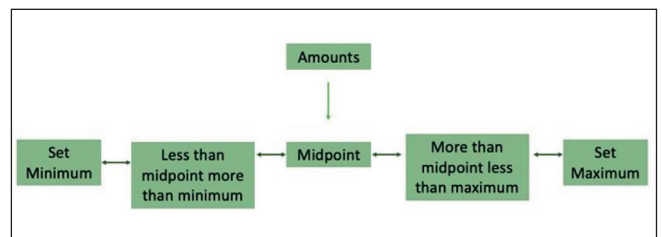
Session 2: A Scratch program to find the smallest number of coins to make a given sum



Session 3: Exploring random and linear search algorithms

Session 4: Solving a binary search jigsaw in Scratch

Session 5: Fixing the buggy Scratch selection sort program



Session 6: Using a quicksort algorithm