3.1.1 Representing algorithms (decomposition)

Lesson plan and printable activities

Pre-requisite knowledge

Students will need to have knowledge of basic computer programming, which should have been covered in KS3.

Materials needed

1. 3.1.1 (Decomposition) Lesson PowerPoint.

   Some of the PowerPoint slides have notes. These can be seen in preview mode.

Starter activity (~5 minutes)

1. **Slide 2:** Students are divided in to pairs and play Guess Who (instructions on slide).

   Alternative: One student could be placed in the centre of the group and the class as a whole get to ask questions.

   Alternative: 20 questions. Guess which famous person I'm thinking of, etc., all provide a similar result.

2. **Slide 3:** Discussion following starter – students asked to think about what questions they asked and how useful they were.

Main activities

Notes: Assessment opportunities are primarily judgement based; teachers will need to apply their own knowledge of the subject matter and the students to decide whether adequate progress is being made. There are few right and wrong answers in this topic, mostly levels of comprehension.

1. **Slide 5:** Introduction of key concepts of decomposition.

2. **Slide 6:** How do we decompose a problem? Without using the word 'recursion' we want students to understand that you repeat the process over and over on each step and sub-step until everything is manageable.

3. **Slide 7:** Whilst a decomposition or hierarchy diagram can be very useful for **visual** learners and are easily hand drawn, it is not easily extendable and can quickly run out of room.

4. **Slides 8 - 10:** Whilst easier to manage using a word processor, lists can be difficult to work with by hand. They do lead on nicely to pseudo-code and/or flowchart planning.

5. **Slide 11:** Decompose the game Rock-Paper-Scissors

   Assessment opportunity: Judge the understanding of students by progress and questions asked. There is no correct answer to this.

6. **Slide 12:** Discussion on which method is easiest.

   Assessment opportunity: Hopefully students will identify the points made in example 3 and 4 above.

7. **Slide 13:** Decompose a recent science experiment – hopefully this will give the students the opportunity to try things that may be a bit different to their neighbour.

   Assessment opportunity: Judge understanding of students and ability to apply abstract thinking.

8. **Slide 14:** Decompose the problem to create a simple calculation program. Students are encouraged to use look for links between decomposed parts and subroutines (and eventually lines) of code.

   Assessment opportunity: Teacher should be circulating and checking progress – if students decompose the problem fully then the program should not be a challenge.

   Alternative: If this is being delivered before students learn to code, then Activity 9 below is an alternative.

9. **Slide 15:** Decompose the problem to create instructions for a board game. If suitable, teacher could offer a single suggestion (eg Monopoly) but the task is dependent on student knowledge. Introduction to breaking code in to subroutines can be discussed.

   Assessment opportunity: Teacher should be circulating and checking progress.

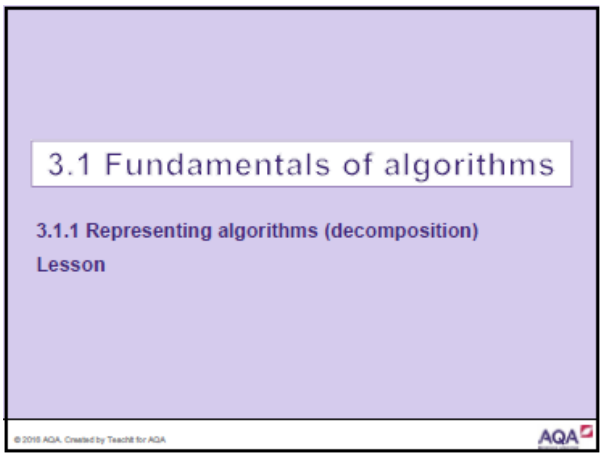
   Alternative: This is an alternative to Activity 8, which may be more engaging if students have the skills.

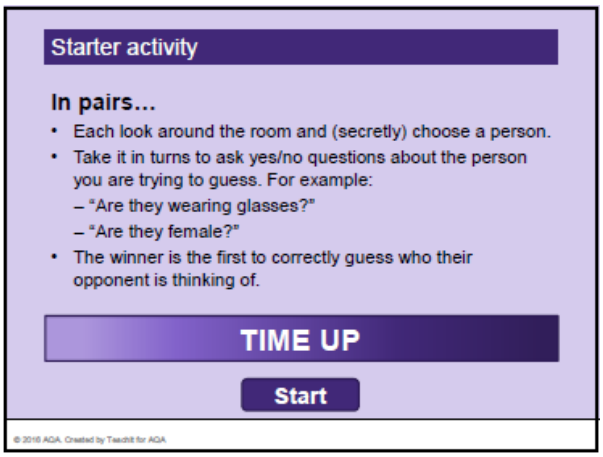
Plenary (5-10 minutes)

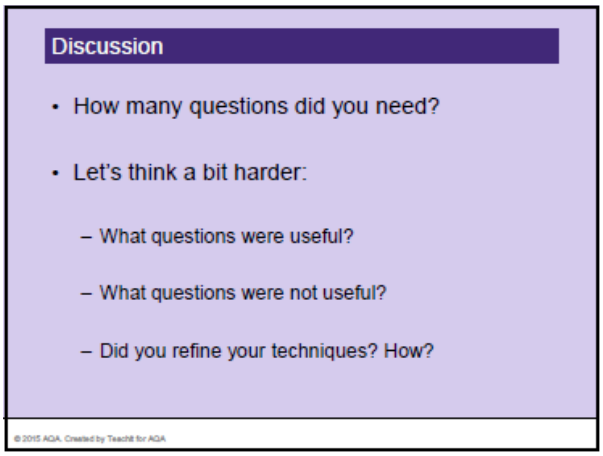
1. **Slide 16:** Groups should be formed:

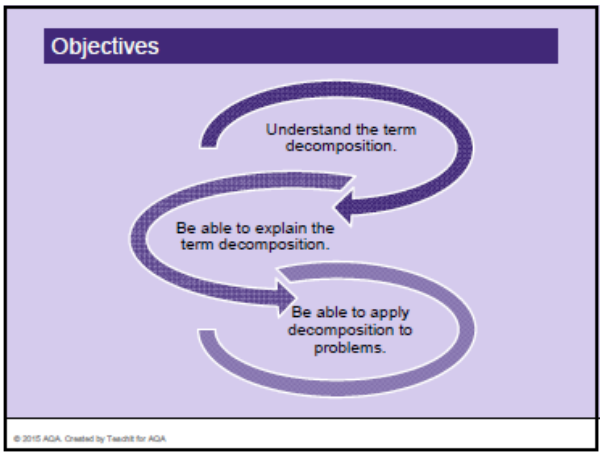
   a. Students who need additional support should be able to form a definition.
   b. The majority of students should be able to write top tips for decomposing a problem.
   c. The best should be able to write a guide.

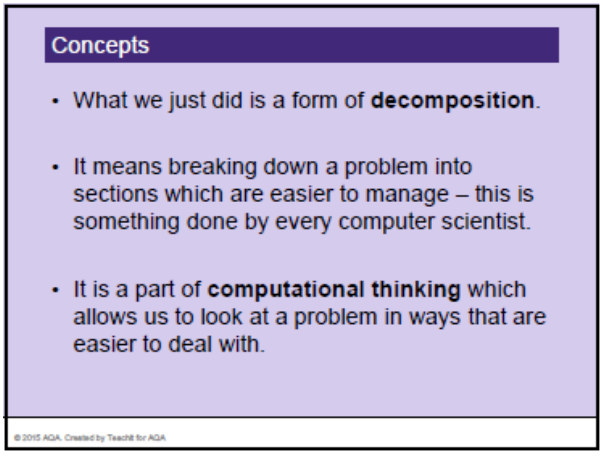
Lesson

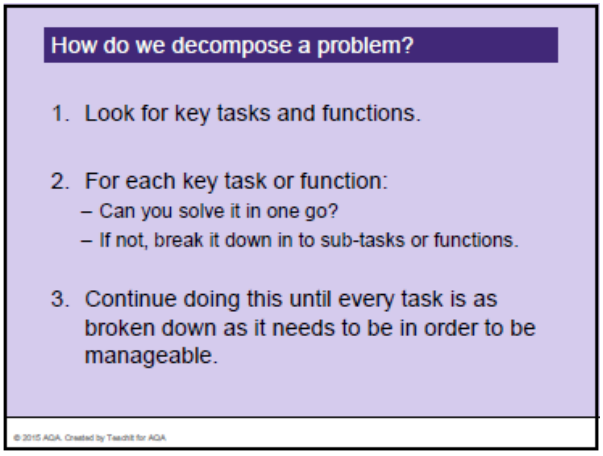








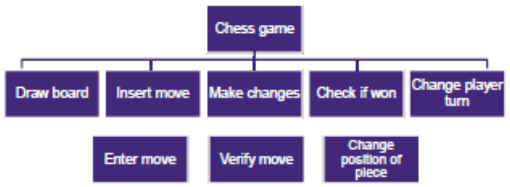




Example

Some people like to use diagrams. Let's say we've been asked to produce a chess game.

We continue breaking down tasks until we are confident we can complete the task as it is.



Example

Some people prefer to use a numbered and sub-numbered list. Let's look at the same game.

Primary tasks are listed

Chess game

1. Draw board
2. Insert move
3. Make changes
4. Check if won
5. Change player turn

Example

Some people prefer to use a numbered and sub-numbered list. Let's look at the same game.

Chess game

1. Draw board
2. Insert move
    1. Enter move
    2. Verify move
    3. Change position of piece
3. Make changes
4. Check if won
5. Change player turn

Example

Some people prefer to use a numbered and sub-numbered list. Let's look at the same game.

It is easier to include more steps this way than with a diagram.

Chess game

1. Draw board
2. Insert move
    1. Enter move
        1. Enter x positon of piece to move
        2. Enter y position of piece to move
        3. Enter x position to move to
        4. Enter y position to move to
    2. Verify move
    3. Change position of piece
3. Make changes
4. Check if won
5. Change player turn

We can refer to these by a dotted number – e.g. 2.1.3

This kind of plan can easily be converted to pseudocode.

Task

- Create a game of rock, paper, scissors.
- **Decompose** the problem to manageable steps:

1. Create a diagram.
2. Create a numbered list of steps.

Discussion

- Which method was easier?
- Why?
- What are the strengths of diagrams?
- What are the strengths of the numbered list?

Task

- Did you notice that we've not actually discussed computers?
- Although these methods are excellent for developing computer programs and inherent in good design work, they can also be used in offline tasks, such as:
  - Planning books or papers – break down to chapters, sections, paragraphs.
  - Building a house – break down to frame, cladding, roof, interior, and further to electrics, plumbing, fittings, décor.
- Think about a recent experiment you did in science – given that your teacher probably gave you an overview, take that overview and decompose it.

Activity

Problem: Write a program that will ask you to choose from a menu to either add, subtract, multiply or delete. Then enter two numbers and apply the operator to them.

Activity: **Decompose** the problem.

Target: Each step in the **decomposition** should take only 1-2 lines of code each.

Bonus: Use the major steps to plan subroutines.

Catch: You can only go to the computer when your teacher is satisfied with the work you have done.

Activity

Problem: Your teacher has set up a board game club after school, but all the games they bought from eBay have come without instructions! You need to write some.

Activity: **Decompose** the problem.

Target: Your teacher or partner should be able to read your instruction booklet and should **not** be able to say "but what if…"

Tip: Use the major steps to plan areas of the instruction booklet.

Plenary

In groups…

- Write a definition of **decomposition**.
- Write a series of top tips for anyone looking to **decompose** a problem.

Extension: Formulate your top tips in to a guide for performing **decomposition** on a problem.