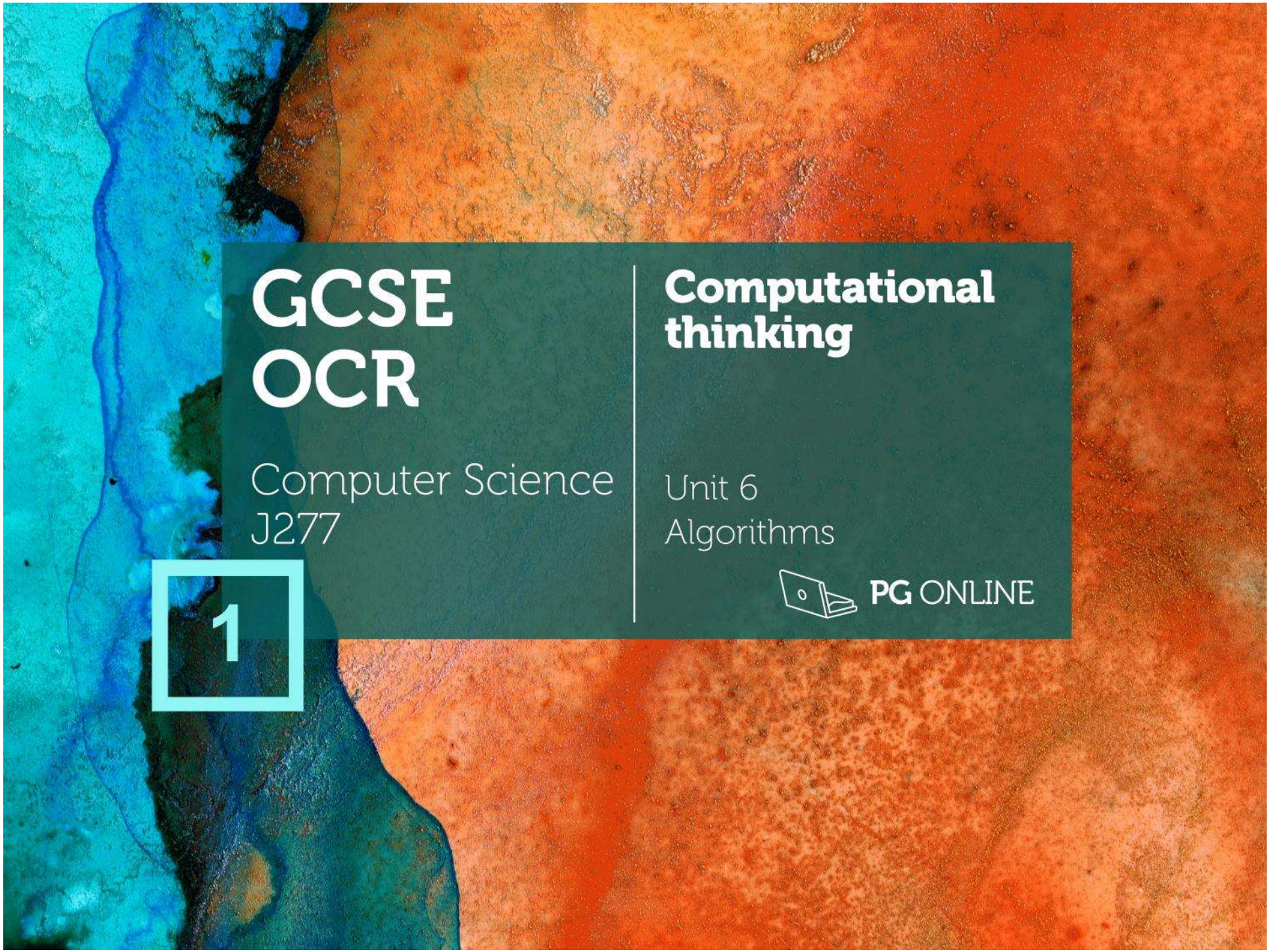

GCSE
OCR
Computer Science
J277
Computational thinking
Unit 6
Algorithms
1
PG ONLINE

# Objectives

- Understand the principles of computational thinking including
  - Abstraction
  - Decomposition
  - Algorithmic thinking
- Be able to produce structure diagrams to show:
  - The structure of a problem
  - Subsections and their links to other subsections

# Starter

- Computer Science is about studying problems and how to solve them
- Discuss some solutions to the following:
  - How can you route pieces of information across a network to the other side of the world?
  - How can you make the images in a computer game look more realistic?
  - How can you program a computer to work out the school timetable?
  - How can you search 1,000,000 items quickly?

# Starter

Answers

- The answers to these problems are too long to write here – some aspects that you may have considered in your discussions?
  - Some problems such as ‘how can you search 1,000,000 items quickly?’ require a specific **algorithm** to be considered
  - There may be many different algorithms that solve the same problem
  - Problems like ‘How can you make the images in a computer game look more realistic?’ need **abstractions**, where we consider the important components of a realistic image
  - Routing items of data around the world requires many different protocols. This is a result of **decomposing** the problem into smaller tasks

# What is an algorithm?

# Algorithms

- An algorithm is a set of instructions for solving a problem or completing a task
- The task could be:
  - Making a chocolate cake
  - Summing the numbers 1 to 1000
  - Building a Lego model
  - Think of some more …

# Strategies for problem-solving

- One strategy for solving a large problem is to first try and solve a similar but smaller problem
  - How do you set about doing a jigsaw puzzle?

# Algorithmic thinking

- Solving these puzzles involves algorithmic thinking
- If you are using a computer to find the answer, you have to figure out how to solve the problem, and then write down the steps
  - Not all solutions are equally efficient

# Divide and conquer

- Here is a problem
- Ask a friend to think of a number between 1 and 1000
- Guess the number by asking: "Is the number greater than ***n***" (where ***n*** is your guess)
  - How many guesses will you need to find the number?

# Worst case scenario

Answers

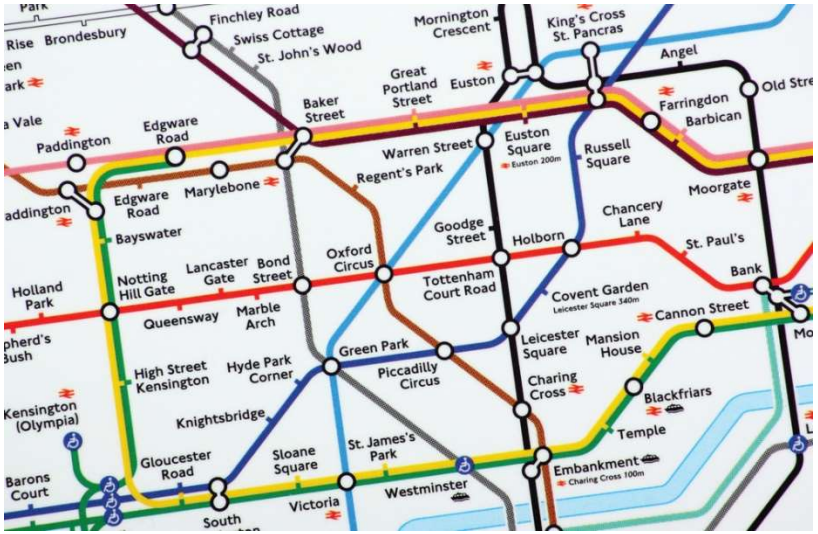
- Search for the number at the mid-point (500)
  - If it's the number we are searching for then stop
  - If it's lower than 500, then search at the mid point of the lower numbers (250)
  - Otherwise, if it's higher than 500, then search at the mid point of the higher numbers (750)
- The worst case scenario for this algorithm would be a search for numbers such as '1' or '367'
  - With this algorithm, any number can be guessed within 10 guesses
  - 1 = 500,250,125,63,32,16,8,4,2,1
  - 367 = 500,250,375,312,344,360,368,364,366,367

# Worksheet 1

- Now complete **Task 1** on **Worksheet 1**

# Abstraction

- Abstraction involves removing unnecessary detail from a problem so that you can focus on the essential components
  - The London Underground map is a good example of abstraction




PG ONLINE

# Abstraction

- When you write a program to play a game involving dice with a computer, how does the computer ‘roll the dice’?

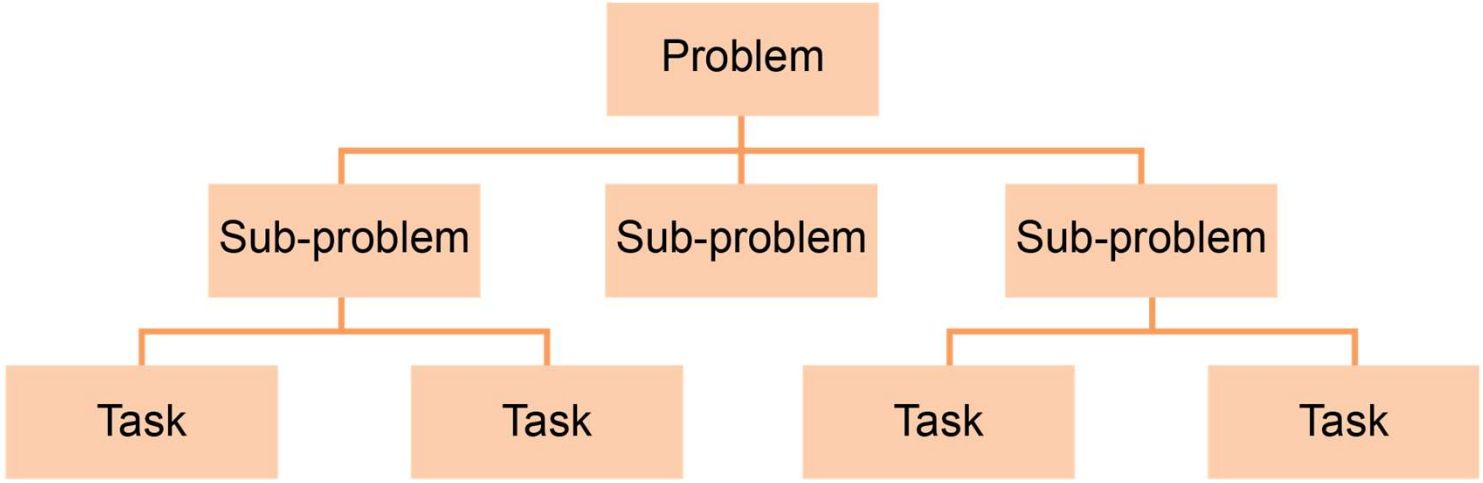
# Rolling dice

Answers

- When a computer rolls a dice we can use abstraction to remove many unnecessary details
  - It will depend on the problem being solved as to what is and isn't important
  - A computer game may need to show a graphical representation of a dice – but they may be able to abstract away all the details about the surface it rolls onto and the physics of the bounce
  - Many programs just need a random number – in which case they don't need to worry about how the dice appears, its weight or how the spots are arranged – they can just find a random number with one line of programming code

# Decomposition

- Decomposition involves breaking down a large problem into smaller sub-problems
- Then the sub-problems can be broken down further until each small task is manageable

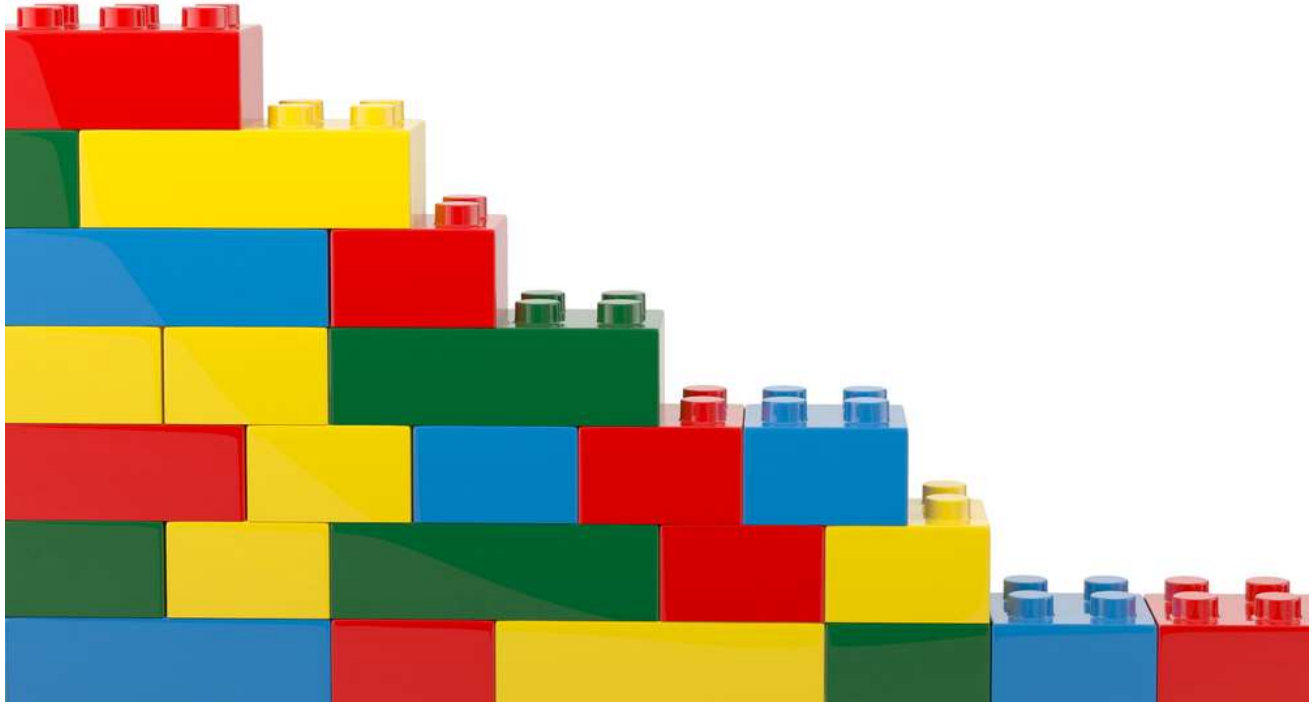


# Decomposing a dice game

- Suppose you want to create a dice game to be played on the computer
- You need to think of the main tasks that need to be performed – for example:
  - Display the rules
  - Computer AI for a two player game
  - Display the board
  - Play the game
  - A results table

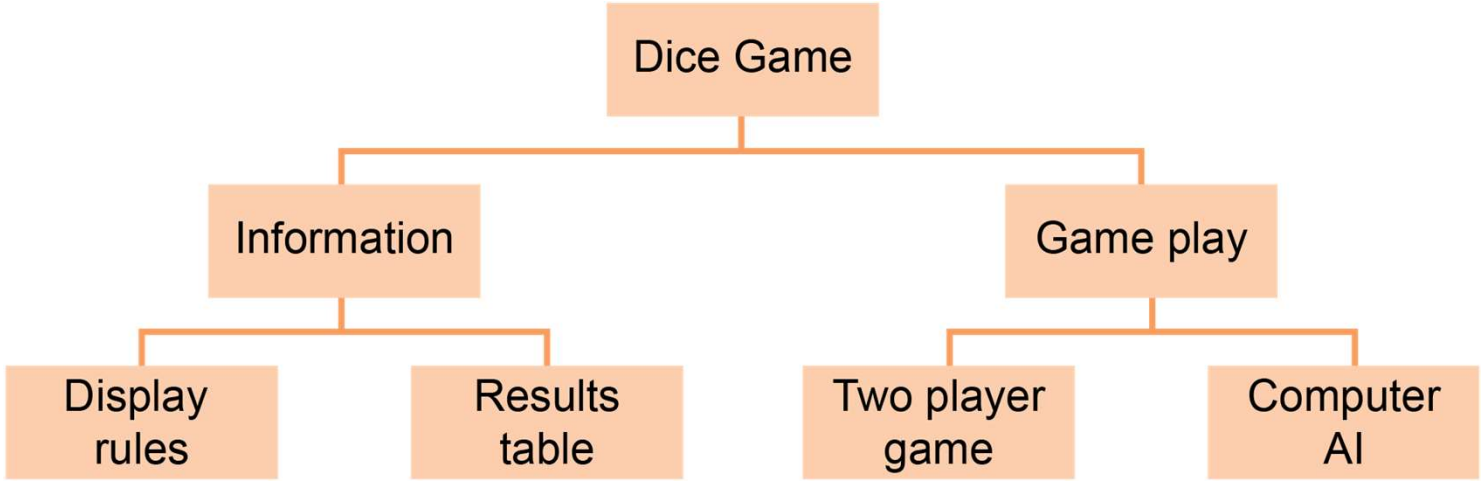
# Decomposition - advantages

- The problem becomes easier to solve when it consists of a number of small subtasks or modules
- Some modules may be reusable in other programs, saving development time



# Structure diagrams

- A structure diagram is used to show how a problem is broken down
  - It will show subsections and their links to other subsections



# Worksheet 1

- Now complete **Task 2** on **Worksheet 1**

# Plenary

- With a partner define the following terms:
  - Abstraction
  - Decomposition
  - Algorithmic thinking
  - Structure diagram

# Plenary

Answers

- Definitions
  - Abstraction – removing unimportant parts of a problem in order to concentrate on those that are important
  - Decomposition – breaking down a problem into smaller more manageable ones
  - Algorithmic thinking – an approach to solving problems by the use of algorithms (sequences of steps that lead to a solution)
  - Structure diagram – a hierarchical diagram that shows how a problem is broken down into sub-sections/sub-tasks

## Copyright



## Licence agreement