LESSON 11

Power of Computing in Solutions

40 minutes | Foundations of Computer Science | Outcome 1.2 | Outcome 1.3 | Outcome 1.6 | Outcome 1.8 | Outcome 1.9 | Outcome 1.12 | Outcome 1.14

WHAT THIS LESSON DOES

In this lesson, you'll uncover how computing transforms complex, time-consuming problems into manageable solutions. Explore the speed, automation, and data-processing power of computers, and apply your understanding through real-world examples and a practical problem-solving activity.

CURRICULUM OUTCOMES

Outcome	What the specification asks for
1.2	explain how the power of computing enables different solutions to difficult problems
1.3	solve problems by deconstructing them into smaller units using a systematic approach in an iterative fashion
1.6	explain the operation of a variety of algorithms
1.8	evaluate the costs and benefits of the use of computing technology in automating processes
1.9	use modelling and simulation in relevant situations
1.12	compare the positive and negative impacts of computing on culture and society
1.14	explain when and what machine learning and AI algorithms might be used in certain contexts

WHAT STUDENTS SHOULD BE ABLE TO DO

- Recognise the characteristics of difficult problems, including scale, complexity, speed, and repetition.
- Understand how computing leverages speed and efficiency to solve challenging tasks.
- Explore methods for handling large data and complex scenarios through computing tools.
- Identify real-world applications of computing in addressing significant problems.
- Apply critical thinking to analyse a problem and propose a computing-based solution.

BEFORE THE LESSON

- No environment or accounts needed: this is a discussion and analysis lesson. Have a few concrete Irish examples ready, such as Met Eireann forecasting or optimising Dublin traffic lights, so the abstract points land.
- Decide how students will capture the practical analysis: spoken pairs, a shared board, or their usual notes. The task asks for pseudocode, so remind them what your agreed pseudocode conventions look like.

Power of Computing in Solutions

HOW THE LESSON RUNS

- Introduction
- What Makes Problems Difficult?
- The Power of Computing: Speed and Efficiency
- Handling Large Data and Complexity
- Real-World Examples
- Practical Activity: Analyse a Problem
- Wrap Up

TEACHING MOVES

- 2. What Makes Problems Difficult? Anchor each characteristic to a number students can feel. Ask how long it would take one person to check every voter in a constituency by hand, then let scale speak for itself. Draw out that speed and error are linked.
- 3. The Power of Computing: Speed and Efficiency. Make parallel processing physical. Have the class add a long list of numbers as one queue, then split into groups doing sections at once. The point is dividing work, not raw speed alone.
- 4. Handling Large Data and Complexity. Keep data structures light here, this is not yet an arrays lesson. Focus on the idea that organising data well is what makes fast access possible, and that patterns come from structure.
- 5. Real-World Examples. For each example, name which of the four characteristics it fights: forecasting is scale plus speed, medical imaging is repetition plus complexity. Ask students to classify a fifth example themselves.
- 6. Practical Activity: Analyse a Problem. Insist they name the difficulty before proposing a solution. Have them say their problem and its characteristic aloud to a partner before writing any pseudocode, so the solution answers a real difficulty.

WHAT TO WATCH FOR

- Students assume computers are simply fast humans and would eventually solve anything given more time, missing that some problems are hard because of scale or complexity, not just clock speed. Contrast a task made easy by speed with one made easy by clever organisation or dividing work. Show that faster alone does not always help.
- Students treat machine learning and algorithms as magic that produces correct answers, rather than tools that find patterns and can be wrong. Point at the medical imaging example and ask what happens when the prediction is mistaken. Establish that a computing solution needs checking, not blind trust.

DIFFERENTIATION

- Emerging: Give them the four characteristics as a checklist and one worked example to copy the structure from before they choose their own problem. Let them describe the problem in plain English and skip pseudocode; the reasoning matters more than the syntax at this stage.