LESSON 25

Algorithms & Efficiency

40 minutes | Hands-On Logic Challenges

WHAT THIS LESSON DOES

In this lesson, you'll explore algorithms by programming a classmate to navigate a grid from start to goal, avoiding obstacles. Through hands-on activities, you'll compare different approaches, test efficiency, and learn the importance of clear, precise instructions.

WHAT STUDENTS SHOULD BE ABLE TO DO

- Understand the concept of algorithms as a set of precise instructions to solve a problem.
- Develop skills in creating clear and efficient sequences of commands.
- Analyse and compare different algorithms for the same task based on efficiency.
- Recognise the importance of debugging and refining instructions to improve outcomes.
- Apply logical reasoning to optimise steps and reduce errors in problem-solving.

BEFORE THE LESSON

- No devices needed: this is an unplugged lesson.
- Mark out a 5x5 floor grid before class using masking tape, floor tiles, or chalk and hula hoops outside. You need enough clear space for pupils to stand around the edge.
- Have a chair or object ready to act as the goal, and a board to write commands and the V-N-O checklist.
- Choose a few obstacle positions on the grid so teams have something to route around.

Algorithms & Efficiency

HOW THE LESSON RUNS

Introduction
Grid Setup: Creating Your Program's World
Student Robot
Precision Programming
Game Time: Direction Dash!
Modelling the Program
Team Challenge: Program a Path
Testing & Debugging: Running Your Program
Retrospective

TEACHING MOVES

- 1. Introduction. Frame the whole lesson as programming: pupils write instructions, a classmate is the robot that runs them. Stress there are many ways to solve the same path, and the point is comparing them.
- 3. Student Robot. Write both commands on the board and act it out live. Let the robot genuinely wander on 'go over there' so the class sees why vagueness fails before you correct it.
- 4. Precision Programming. Draw V-N-O on the board and leave it up all lesson. Point to it and make every command name a clear verb, a number with units, and one instruction per line in order.
- 5. Game Time: Direction Dash! Keep the command set small: TURN Left/Right 90, FORWARD 1 or 2, BACK 1, PAUSE. Call one command every few seconds and have pupils echo it back before moving so nobody guesses.
- 6. Modelling the Program. Name the sequence a program or algorithm as you build it. Give one precise instruction at a time to the robot so the class sees a full program run from start square to chair.
- 7. Team Challenge: Program a Path. Set groups of three or four and assign Planner, Robot and Tester. Rotate the roles between runs so every pupil programs, executes and checks at least once.
- 8. Testing & Debugging: Running Your Program. Cycle one team onto the grid at a time. Remind the robot to run the script exactly with no improvising, so a bug shows plainly and the team can fix the instruction, not the robot.
- 9. Retrospective. Ask each team for one improvement that shortens or clarifies their program, such as a REPEAT for a repeated turn or combining two turns into 180. Keep these spoken, not written.

WHAT TO WATCH FOR

- Pupils blame the robot when a program fails, telling the child to just walk to the goal rather than fixing the instructions. Insist the robot only ever runs the written script. Send the team back to their instructions to find and rewrite the faulty line, showing that the bug lives in the code.
- Pupils leave out the number or the unit, writing 'FORWARD' or 'TURN' and expecting the robot to know how far. Point to the V-N-O checklist and refuse to run any line missing a number. Have them add the exact squares or degrees before testing.
- Pupils assume the robot faces the same way after a turn as it did before, so a later FORWARD goes the wrong direction. Pause after each turn and ask the class which way the robot is now facing before the next command runs.

DIFFERENTIATION

- Emerging: Give this pupil the Robot role first so they experience commands physically before writing any. Shorten their target: a straight three-square path with no obstacles to build confidence.
- Developing: Have them read their program aloud one line at a time and predict where the robot lands before it moves. Pair them as co-Planners so they talk each instruction through together.