

Week 2 Seminar: Search

Your maze pseudocode, search in the real world, and the “best” route | **Solutions**

Work in groups of 3 or 4. This sheet is **not assessed**. Activity 1 is a 5-minute recap at the start of the session; Activities 2 and 3 follow the lecture (about 15 minutes).

Activity 1 | Remember Week 1?

5 minutes, start of session

Get out your group’s pseudocode for the Week 1 maze puzzle.

- (a) When your program had several squares it could try next, which one did it pick?
- (b) After today’s lecture: is your pseudocode depth-first search, breadth-first search, or something else?

Answers

- (a) Common answers: “the one we just found” (the newest), “the one we found first” (the oldest), or “the one closest to B”.
- (b) Newest first is depth-first search: the frontier is a *stack*. Oldest first is breadth-first search: the frontier is a *queue*, and it finds the shortest path. “Closest to B first” is greedy best-first search, using a heuristic (Week 3). “Keep a hand on the wall” is a fixed rule, not a search: it does not keep a frontier and fails in some mazes.

Come back to this after the BFS slides if time is short at the start.

Activity 2 | Search in the real world

10 minutes

For each system, describe it as a search problem.

	States	Actions	Goal	Path cost
A sat-nav				
A chess engine				
A warehouse robot				

- (a) Which of the three is different from the others, and why does that need a different algorithm?

Answers

	Sat-nav	Chess engine
States	Positions on the road map (junctions)	Board positions
Actions	Drive along a road to the next junction	Legal moves
Goal	Arrive at the destination	Checkmate the opponent
Path cost	Travel time (or distance, or fuel)	Not really a cost: what matters is winning

Warehouse robot: states are the robot's position (and what it is carrying); actions are moving one square, picking up or putting down; the goal is the item at the packing desk; the path cost is time or distance, while avoiding other robots.

- (a) The chess engine: there is an opponent who moves too. That is *adversarial search*, so it uses minimax with alpha-beta pruning, and an evaluation function because the game tree is far too big to search to the end. A sat-nav uses A* with a heuristic such as straight-line distance (Week 3).

Activity 3 | Discuss: the "best" route

10 minutes

A sat-nav app picks the "best" route for millions of drivers.

- (a) List three different ways to measure the cost of a route.
- (b) For each one, who gains and who loses?
- (c) Who should decide which cost the app uses by default: the company, the driver, or someone else?

Answers

- (a) Travel time; distance; fuel or emissions; avoiding tolls or motorways; safety (for example avoiding narrow roads).
- (b) Fastest routes suit drivers but can send traffic, including lorries, down residential streets and through quiet villages to avoid a queue. Shortest routes can use roads that are not suitable. Fuel-saving routes help the environment but may be slower.
- (c) The company sets the default, which affects millions of journeys. Drivers can change settings, but most keep the default. Councils can restrict roads (for example weight limits for lorries). Point to draw out: **the path cost function is a design decision with effects on people**, not just a technical detail. Links to learning outcome (d).

Portfolio link

Part 2a of the portfolio asks you to say how a method works on a problem. For a search method, that means naming the states, actions, goal and path cost, as in Activity 2.