2.6 Advanced Algorithms: Searching, Sorting and Path-Finding: 10-minute Pack

OCR H446 · Calculators not allowed · about 10 minutes

Total Marks

Name: _____ Date: ____ / ____ / _____

Answer ALL questions. Show all your working.

1 A satnav app calculates driving routes across a national road network containing millions of junctions.

(a) Discuss the relative advantages of using Dijkstra's algorithm compared with the A* algorithm for calculating a single driving route between two specific junctions in this network, and describe one situation in which Dijkstra's algorithm would still be the more appropriate choice. **(8)**

(Total for Question 1 is 8 marks)

© Revision Library | revisionlibrary.co.uk

Mark scheme · 2.6 Advanced Algorithms: Searching, Sorting and Path-Finding: 10-minute Pack

Question 1

(a) **L1** Basic, isolated points are made about Dijkstra's algorithm and/or A*, with little or no reference to time complexity, heuristics or the given scenario. The answer may simply name both algorithms with little justification, or fail to identify any valid situation where Dijkstra is preferable. **1-3**

(a) **L2** Some relevant points are made about the efficiency of A* compared with Dijkstra, with limited or partly-accurate reference to heuristics and time complexity. Some link is made to the scale of the scenario (millions of junctions), and a situation favouring Dijkstra is attempted but may be only partially justified. **4-6**

(a) **L3** A clear, well-justified discussion is given, correctly explaining how A*'s heuristic reduces the number of junctions explored compared with Dijkstra for a single specific route, supported by accurate technical detail (e.g. admissibility). A valid, clearly justified situation where Dijkstra remains the more appropriate choice is identified and linked to the scenario. **7-8**

(a) Indicative content:

- A* uses a heuristic (e.g. straight-line distance to the destination) to prioritise exploring junctions likely to lie on the route to the specific destination, so across a network of millions of junctions it can typically reach the destination while expanding far fewer vertices than Dijkstra's algorithm, which explores outward in all directions with no sense of direction towards the goal.
- Both algorithms are guaranteed to find the shortest path provided the heuristic used by A* is admissible (never overestimates the true remaining distance); straight-line distance is always less than or equal to the true road distance, so it is a valid admissible heuristic here.
- Dijkstra's algorithm remains the more appropriate choice when the shortest paths from a single junction to every other junction are needed at once (a single-source, all-destinations problem) - for example, precomputing journey times from a delivery depot to every possible drop-off point overnight - because A*'s heuristic advantage only helps when searching towards one specific, known destination.
- Dijkstra may also be preferred when no reliable, cheap-to-compute heuristic is available for the scenario, since a poor heuristic could remove most or all of A*'s efficiency benefit (in the extreme case, a heuristic that is always 0 makes A* behave identically to Dijkstra).
- A stronger answer may note the practical trade-off: A* needs a suitable heuristic (e.g. precomputed junction coordinates) available for every vertex, which adds a small implementation and data overhead that plain Dijkstra does not require.

(a) Answer: Best answers recommend A* for a single specific route, using an admissible heuristic such as straight-line distance to prioritise the search and expand far fewer junctions than Dijkstra, but correctly identify that Dijkstra is preferable when shortest paths to all destinations are needed simultaneously (e.g. depot-to-everywhere precomputation), since A*'s heuristic gives no benefit in that case.

© Revision Library | revisionlibrary.co.uk