Week 2: Path Search (Open leering Week 3)

Tutorial 3: Path Search

3.1 The Missionaries & Cannibals Problem (Activity 3.1)

The "Missionaries and Cannibals" problem is usually stated as follows: Three missionaries and three cannibals are on one side of the river, along with a boat that can hold one or two people. Find a way to get everyone to the other side, without ever leaving a group of missionaries in one place outnumbered by the cannibals in that place. This problem is famous in Artificial Intelligence because it was the subject of the first paper that approached problem formulation from an analytical viewpoint (Amarel, 1968). (You can even play the puzzle on-line at www.learn4good.com/games/puzzle/boat.htm)

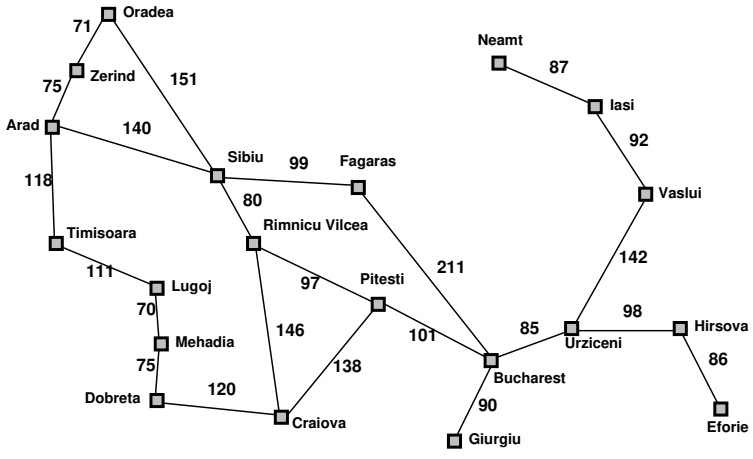
What would be the best way to represent each "state" in the Missionaries and Cannibals problem? How many possible states are there? Make only those distinctions necessary to ensure a valid solution. Draw a diagram of the complete state space.

1. Formulate the problem precisely, making only those distinctions necessary to ensure a valid solution, and draw a diagram of the complete state space.

2. Solve the problem optimally using an appropriate search algorithm; is it a good idea to check for repeated states?

3. Why do you think people have a hard time solving this puzzle, given that the state space is so simple?

3.2. This exercise concerns the route-finding problem using the Romania map from Russell & Norvig (Artificial Intelligence: A Modern Approach) as an example.



Straight−line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Define the route-finding problem (from Arad to Bucharest) as a state space search problem (give short descriptions of the state space, etc. in English). What order are nodes in the state space expanded for each of the following algorithms when searching for a (shortest) path between Arad and Bucharest (when there is a choice of nodes, take the one earliest in the alphabetical ordering)? Make sure you understand the key properties of the different algorithms, as listed below.

To clarify, for breadth-first search, stop the search when the goal state is generated and use a check to ensure that nodes with the same state as a previously expanded node are not added to the frontier. For the other search algorithms, stop the search when the goal state is

expanded; for uniform-cost search include a check that nodes with the same state as a previously expanded nodes are not added to the frontier (as in breadth-first search) and a test so that only one node for a given state is stored on the frontier (that with the shortest path to that state), and for depth-first search use cycle checking along a path to avoid repeated states that may lead to infinite branches.

(i) Depth-first search (efficient use of space but may not terminate)
(ii) Breadth-first search (space inefficient, guaranteed to find a solution)
(iii) Uniform-cost search (similar to breadth-first, but order nodes by cost)

Which algorithm is suitable in practice for solving route-finding problems such as this?

3.3 This version of the map (from the first edition of the book) differs from that in the second edition of the book in that the heuristic value for Fagaras is 178 rather than 176, and that for Pitesti is 98 rather than 100 (also Drobeta is mistakenly spelt as Dobreta). What difference does this make?

Students activity

(Activity 3.2): Refining Your Understanding of Searches using Mazes

Discuss your findings and insights from the 'Fun with Mazes' activity. Compare your findings and discuss any discrepancies.

(a) Generate a random Tree Maze and compare the running time of Breadth First Search and Depth First Search on this maze. Repeat this for a couple of other random Tree Mazes. Time the algorithms with a stopwatch if you can. Which algorithm is faster?

(b) Repeat the steps from part (a), this time with Concentric Graph Mazes. Which algorithm is faster?

(c) Use the widget to edit a maze by clicking on the coloured squares, and then clicking on the Maze. Try to design a Maze for which BFS finds a solution considerably faster than DFS. You can specify a starting and ending point anywhere inside the maze. Try to (briefly) explain in words what makes your environment easy for BFS but hard for DFS.

(d) Use the widget to create a maze for which DFS would find a solution considerably faster than BFS. You should assume that there can be multiple ending points (red squares) and that the algorithm only needs to reach one of them. Try to explain in words what makes your environment easy for DFS but hard for BFS.

(e) Try running Iterative Deepening Search on a random maze. Why is it so slow? For which type of problem (probably not a maze) would IDS be superior to both BFS and DFS?