COMP3411/9414/9814 Artificial Intelligence
Session 1, 2018

Week 3 Tutorial Solutions

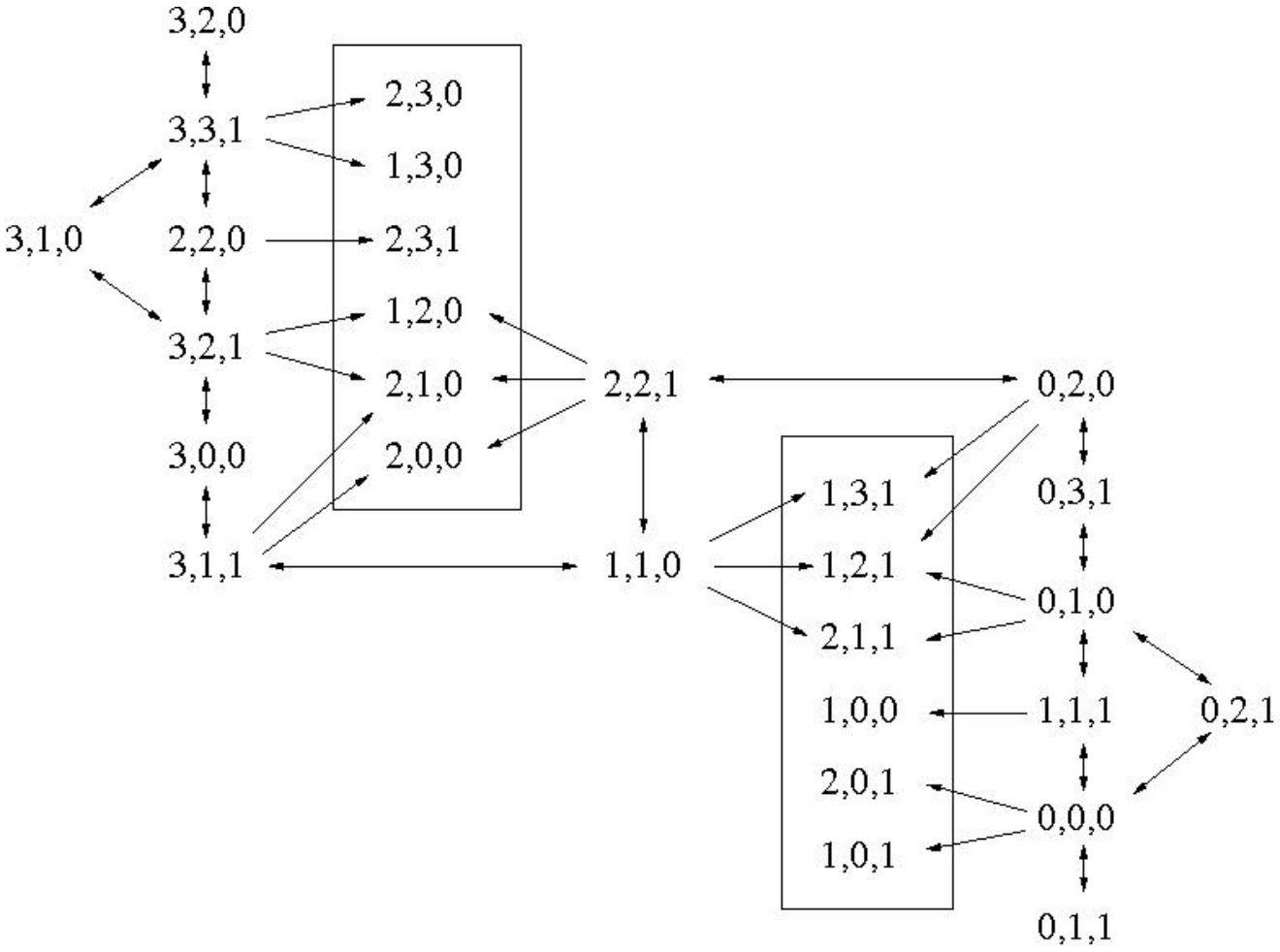
This page was last updated: 03/17/2018 09:51:38

Activity 3.1 (Exercise 3.9 from Russell & Norvig)

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 AI 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")

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

   Each state can be characterized by listing the number of missionaries (0-3) and cannibals (0-3) on the original side of the river, and 1 or 0 to indicate whether or not the boat is on that side; it is assumed that whatever is not listed must be on the far side of the river. There are 4 x 4 x 2 = 32 states in total, but only 28 of them are accessible from the initial state (3,3,1). Twelve of these represent "dead" states in which one or more missionaries are eaten. Here is 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?

There are four ways to get from the initial state (3,3,1) to the final state (0,0,0) in 11 steps. It is definitely a good idea to check for repeated states. Using Breadth-First Search, for example, there are 6 nodes at depth 2 and 25 at depth 3; but, if we avoid expanding previously encountered states, there will only be 2 nodes at depth 2 and 3 at depth 3. Depth-First Search is only able to avoid states which are repeated along the same branch; but the number of nodes is still reduced to 3 at depth 2 and 8 at depth 3.

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

   The step at which people seem to have most difficulty is the one in the centre of the diagram, from (1,1,0) to (2,2,1). Since the objective is to get all 6 people to the far side of the river, it seems counterintuitive to bring two people back to the original side; it violates the "heuristic" of maximizing the total number of people on the far side of the river.

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?

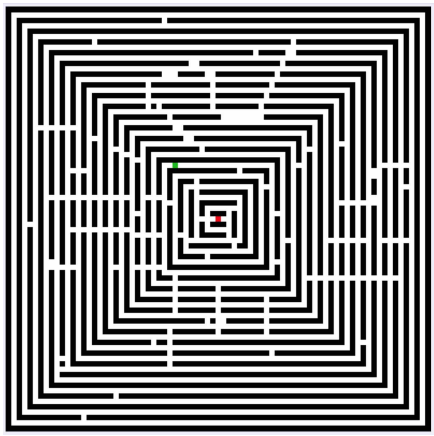
   Generally, BFS is a bit faster.

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

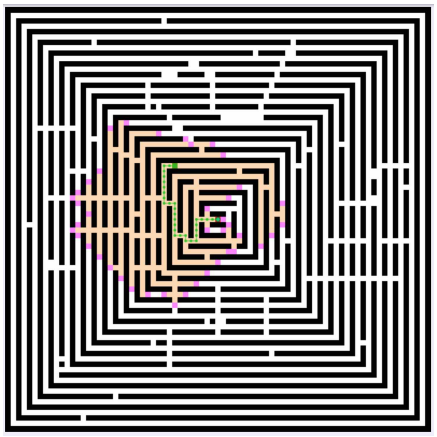
   Depth First Search is faster.

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 change the structure of the maze, and 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.

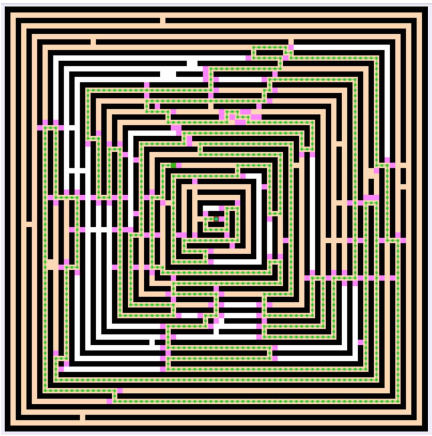
   This would be the case for a fairly open maze, with a single goal which is relatively close to the initial state. BFS will find this goal quickly. DFS may occasionally get lucky, but will often get lost exploring the vast expanses around the goal before finally reaching it.



maze



Breadth First Search



Depth First Search

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.

   This would be true for a search problem where there exist many solutions at a fairly deep level, such that DFS will find a solution more or less as soon as it gets to that depth for the first time. For example, imagine you are shipwrecked in the middle of a lake and you need to

find a path to the shoreline, with a series of small steps either up, down, left or right. DFS will find a path quickly, whereas BFS will spend a lot of time exploring all paths shorter than the radius of the lake.

"lake" maze

Breadth First Search

Depth First Search

e. Try running Iterative Deepening Search on a random maze. Why is it so slow?

Iterative Deepening Search generally performs poorly in situations (like this one) with a low branching factor. Another example is the Missionaries and Cannibals problem. As an extreme example, consider a problem with a branching factor of 1 at every move, and a solution at depth n. Depth-First Search will take exactly n expansions to find a solution, while Iterative Deepening Search will take $O(n^2)$.

For which type of problem (probably not a maze) would IDS be superior to both BFS and DFS?

For problems with a larger branching factor (for example, the 8-Puzzle, with branching factor 4), IDS would be superior to both BFS and DFS (in the sense of finding a solution in reasonable time without running out of memory).

Activity 3.3 Further Discussion

Any remaining tutorial time should be used for further discussion related to uninformed search strategies, including the Pick a Problem and Map out the State Space activity on the Path Search Problems page; for example:

- Additional Mazes and search strategies
- Fox, Goose and Bag of Beans
- Tower of Hanoi
- Bridge and Torch puzzle
- 5-Puzzle

Answers to Romania Path Search Activities:

- Breadth First Search, to find a path from Arad to Bucharest:

  Arad, Sibiu, Timisoara, Zerind, Fagaras.

- Uniform Cost Search, to find a path from Dobreta to Fagaras:

  Dobreta (0), Mehadia (75), Craiova (120), Lugoj(145), Timisoara (256), Pitesti (258), Rimnicu Vilcea (266), Sibiu (346), Bucharest (359), Arad (374), Urziceni (444), Fagaras (445).

- Depth First Search, to find a path from Dobreta to Zerind:

  Dobreta, Craiova, Pitesti, Bucharest, Fagaras, Sibiu, Arad, Timisoara, Lugoj, Mehadia.
