CS3243 Introduction to Artificial Intelligence

AY2021/2022 Semester 1

Tutorial 1: Introduction & Uninformed Search

Important Instructions:

- **Assignment 1** consists of **Question 5** and **Question 6** from this tutorial.
- Your solutions for the above questions may be handwritten or typewritten, but handwritten solutions must be legible for marks to be awarded.
- You are to submit your solutions on LumiNUS by **Week 4 Monday (30 August), 2359 hours**, under **Tutorial Submission** → **Tutorial 1**. If you are submitting handwritten solutions, please append the question paper in front of your solutions.

Note: you may discuss the content of the questions with your classmates, but you must work out and write up your solution individually - solutions that are plagiarised will be heavily penalised.

Tutorial Questions

(1) Sudoku is a popular number puzzle that works as follows: we are given a 9×9 square grid; some squares have numbers, while some are blank. Our objective is to fill in the blanks with numbers from $1 - 9$ such that each row, column, and the highlighted 3×3 squares contain no duplicate entries (see Figure 1.1).

Sudoku puzzles can be solved easily after being modelled as a CSP (which will be covered later in the course). We will consider the problem of *generating* Sudoku puzzles. In particular, consider the following procedure: start with a completely full number grid (see Figure 1.2), and iteratively make some squares blank. We continue blanking out squares as long as the resulting puzzle can be completed in at least one way.

2	5			3		9		1
	1				4			
4		7				2		8
		5	2					
				9	8	1		
	4				3			
			3	6			7	2
	7							3
9		3				6		4

Figure 1.1. A simple Sudoku puzzle.

2	5	8	7	3	6	9	4	1
6	1	9	8	2	4	3	5	7
4	3	7	9	1	5	2	6	8
3	9	5	2	7	1	4	8	6
7	6	2	4	9	8	1	3	5
8	4	1	6	5	3	7	2	9
1	8	4	3	6	9	5	7	2
5	7	6	1	4	2	8	9	3
9	2	3	5	8	7	6	1	4

FIGURE 1.2. Solution to the puzzle in Figure 1.1.

Complete the following:

- Give the representation of a state in this problem.
- Using the state representation defined above, specify the initial state and goal state(s).
- Define its actions.
- Using the state representation and actions defined above, specify the transition function T. (In other words, when each of the actions defined above is applied to a current state, what is the resulting state?)

Solution:

- A state in this problem is a (partial) valid solution of the Sudoku puzzle. More formally, it would be a matrix $M \in \{0, \ldots, 9\}^{9\times 9}$, with 0 representing a blank square.
- The initial state is a completed filled out grid of numbers which is a valid. All rows, columns, and 3×3 squares should contain all numbers between $1, \ldots, 9$. Any state in the problem will be a valid goal state.
- An action would be removing a number from the grid. More formally, we take as input a matrix M and a matrix $E_{i,j}(a)$, where $E_{i,j}(a) \in \{0, \ldots, 9\}^{9\times 9}$ is a matrix of all zeroes except for a nonzero value $a \in \{1, \ldots, 9\}$ in coordinate (i, j). An action would be setting $M - E_{i,j}(a)$.
- The transition model would be $T(M, E_{i,j}(a)) = M - E_{i,j}(a)$.

  (It is worthwhile to think about how would you make a *good* puzzle generator. It would probably make sense to randomly remove numbers rather than deterministically doing so. There are many goal nodes here, but it would probably be boring to use your Sudoku generator if it always outputed the same puzzle on the same input.)

(2) (a) Describe the difference between the **Tree Search** and **Graph Search** algorithms.

Solution: With the tree-search implementation, we may revisit states, but with the graph-based implementation, we maintain an explored set to ensure that we do not revisit states.

(b) Assuming that ties (when pushing to the frontier) are broken based on alphabetical order, specify the order of the nodes that would be explored by the following algorithms. Assume that S is the initial node, while G is the goal node.

You should express your answer in the form $S - B - A - F - G$ (i.e. no spaces, all uppercase letters, delimited by the dash ($-$) character), which, for example, corresponds to the exploration order of S, B, A, F, then G.

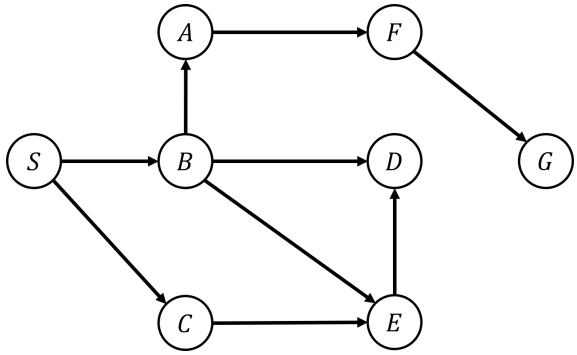

FIGURE 1.3. Graph for question 2(b).

(i) Depth-First Search with tree-based implementation.

(ii) Depth-First Search with graph-based implementation.

(iii) Breadth-First Search with tree-based implementation.

(iv) Breadth-First Search with graph-based implementation.

Solution: (assume there are no duplicates in the frontier)

(i) $S - C - E - D - B - E - D - A - F - G$

(ii) $S - C - E - D - B - A - F - G$

(iii) $S - B - C - A - D - E - F - G$
(assuming GoalTest is performed within the **for** loop).
Note: this solution is based on the algorithm given in the lecture.[1]

(iv) $S - B - C - A - D - E - F - G$

(3) (a) The **Breadth-First Search** algorithm is complete if the state space has infinite depth but finite branching factor.

Determine if the statement above is True or False, and provide a rationale.

Solution: This is true. This follows immediately from the definitions of Breadth-First Search and completeness. (Recall that a search algorithm is complete if whenever there is a path from the initial state to the goal, the algorithm will find it.)

In particular, the existence of a path means that the goal exists in a finite depth, and therefore Breadth-First Search will eventually reach it, since having a finite branching factor implies that the algorithm wouldn't get lost in infinite-breadth search at some level.

[1] If GoalTest is performed outside the **for** loop (as per the case of UCS), the solution would be $S - B - C - A - D - E - F - D - G$. Note that for exams, you should consult the algorithms provided in the lecture notes.

(b) The **Breadth-First Search** algorithm can be complete even if zero step costs are allowed.

Determine if the statement above is True or False, and provide a rationale.

Solution: This is true. As highlighted in the previous question, Breadth-First Search is complete as long as the state space has a finite branching factor. Note that Breadth-First Search (as well as Depth-First Search) does not take into account edge weights.

(c) Given that a goal exists within a finite search space, the **Breadth-First Search** algorithm is optimal if all step costs from the initial state are non-decreasing in the depth of the search tree. That is, for any given level of the search tree, all step costs are greater than the step costs in the previous level.

Determine if the statement above is True or False, and provide a rationale.

Solution: This is false. We could have two goal nodes in the same level with different costs but with the same depth. For example,

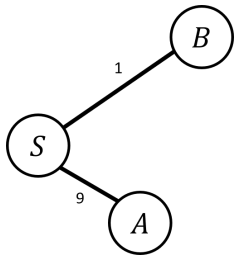

FIGURE 1.4. A counterexample for question 3.

Arbitrary tie-breaking (or breaking ties by alphabetical ordering) of the Breadth-First Search algorithm may result in A reached before B.

(4) Prove that the **Uniform Cost Search** algorithm is optimal as long as each step cost exceeds some small positive constant ϵ.

Solution: Given that each step cost exceeds some small positive constant ϵ, completeness may be assumed. Consequently:

- Whenever UCS expands a node n, the optimal path to that node has been found. If this was not the case, there would have to be another frontier node n' on the optimal path from the start node to n. By definition, n' would have a lower value of g than n, and thus would have been selected first.
- If step costs are nonnegative, paths never get shorter as nodes are added.

The above two points together imply that UCS expands nodes in the order of their optimal path cost.

Assignment Questions

(5) You are given an n-piece unassembled jigsaw puzzle set (you may assume that each jigsaw piece can be properly connected to either 2, 3, or 4 pieces), which assembles into an $(m \times k)$ rectangle (i.e. $n = m \times k$). There may be multiple valid final configurations of the puzzle, the picture illustrates an example.

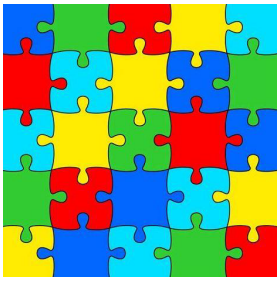

Figure 1.5. A sample configuration of the jigsaw puzzle.

Formulate the above as a search problem. More specifically, define the following:

- State representation
- Initial state
- Actions
- Transition model
- Step cost
- Goal test

If necessary, you may identify the assumptions you have made. However, assumptions that are contradictory to any instructions in the question, or that are unreasonable, will be invalid.

Solution:

- State representation:
  - Keeps track of the puzzle pieces that are connected/unconnected.
  - Keeps track of the connections between connected puzzle pieces.
- Initial state:
  - Depending on the representation, either the set of connected puzzles is empty, or the set of unconnected puzzle pieces is full. There should also be no connections between puzzles initially.
- Actions:
  - Taking two unconnected puzzle pieces and establishing a connection between them. Note that you much mention that the pair of puzzle pieces picked can be legally connected (i.e. cannot just take *any two* puzzle pieces).
- Transition model:
  - Depending on the representation, can either add to the connected set or remove from the unconnected set.

- Step cost:
  - Step costs are 1 (or any positive constant value, cannot be 0).
- Goal test:
  - Depending on the representation, either the connected set is full, or the unconnected set is empty.
  - Check that for every puzzle piece, there should not be more connections than its number of available legal connections.

(6) You have just moved to a strange new city, and you are trying to learn your way around. Most importantly, you want to learn how to get from your home at S to the subway at G.

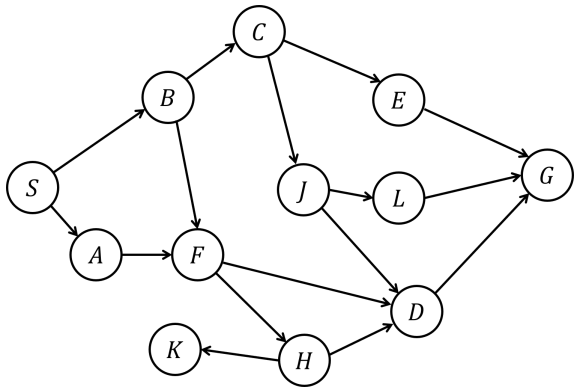

FIGURE 1.6. Graphical representation of the city.

In the following problems, use alphabetical order to break ties when deciding the priority for pushing nodes into the frontier.

(a) Using **Depth-First Search with tree-based implementation**, draw the part of the search tree that is explored by the search.

(b) What is the final path found from the start (S) to the goal (G)? Note that you **MUST** express your answer in the form $S-B-C-J-L-G$ (i.e. no spaces, all uppercase letters, delimited by the dash ($-$) character), which, for example, corresponds to the exploration order of S, B, C, J, L, then G.

Solution: (assume there are no duplicates in the frontier)

(a)

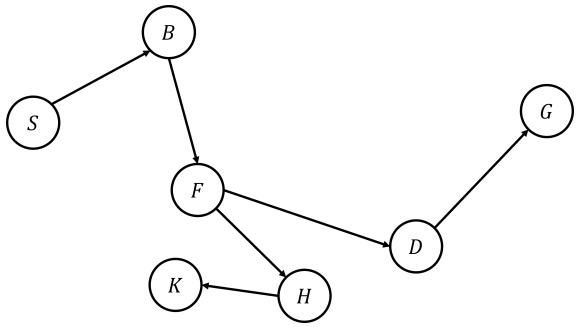

FIGURE 1.7. Solution for question 6(a).

(b) $S-B-F-D-G$