COMP-424: Artificial intelligence

Homework 1

Due on *myCourses* Jan 29, 11:59pm.

General instructions. You must read these carefully before starting!

Course material for answering questions 3 & 4 will be taught next week (Jan. 21, 23).

As explained in class, due to the larger number of students, and relatively few TAs, only 1 or 2 questions out of 5 will be marked in detail. All other questions will be marked based on a few specific answers. A detailed solution set for all questions will be provided after the deadline. It is your responsibility to make sure you understand the solution to all problems.

Your written submission should respect the following format:

- Page 1 should contain a filled answer sheet in the format shown below. Each line should contain the answer to the portions of the questions that are underlined. This will usually be a single word, character, symbol or number. *E.g. For 1(a), you should write down the number of states.*
- Remaining pages should show your full work for all questions. This is very important to include, as 1 or 2 of these will be marked, and require a detailed answer to achieve full marks.
- Submit a single pdf document containing all your pages to McGill's *myCourses*. You can scan-in hand-written sheets for pages 2 and beyond. There are many ways to combine many pdf files into one: learn how to do this!
- Submit your code for question 5 as a separate file to McGill's *myCourses*. Instructions on this (below) have been updated as of Jan.17, 1:30pm. Your code should accept an input file in the format specified below. Sample code is provided on the course website.

Sample answer sheet. You should copy/paste this for the first page of your answer set, or re-produce.

Name: ______________________________________

Student id: __________________________________

1(a) ______

1(b) ______

2(a) (i) ______ (ii) ______ (iii) ______ (iv) ______

2(b) (i) ______ (ii) ______ (iii) ______ (iv) ______

3(a) ______

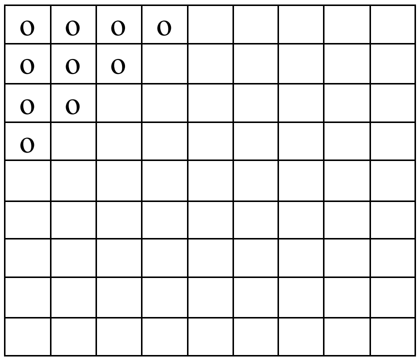
3(b) ______

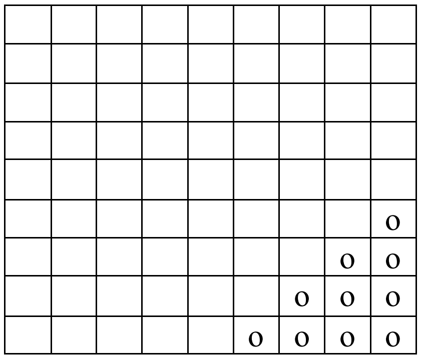
4(a) (i) ______ (ii) ______

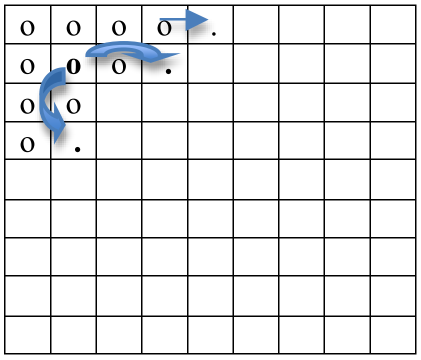
4(e) ______

Question 1: Uninformed search

Consider a 9x9 game board, with 10 identical pieces, as shown in the picture. Pieces can move in two different ways: move to an empty adjacent square in the 4 cardinal directions (shown in panel (iii) for the top right most piece), or jump over an adjacent piece to the square directly opposite of that piece (assuming that square was empty). The goal is to move all of your pieces from one corner to the other corner, using as few moves as possible.


(i) Start configuration


(ii) Goal configuration


(iii) Example move

(a) Describe a state space for this puzzle. How many states are there? If necessary, give an upper bound.
(b) What is the branching factor for this problem? Give the maximum branching for any state and explain your solution.
(c) What is the maximum number of moves required to reach a solution? Give an upper bound (as tight as possible) and explain your solution.
(d) Identify a suitable uninformed search algorithm for this task and justify your choice based on the properties of the problem domain.

Question 2: Informed search

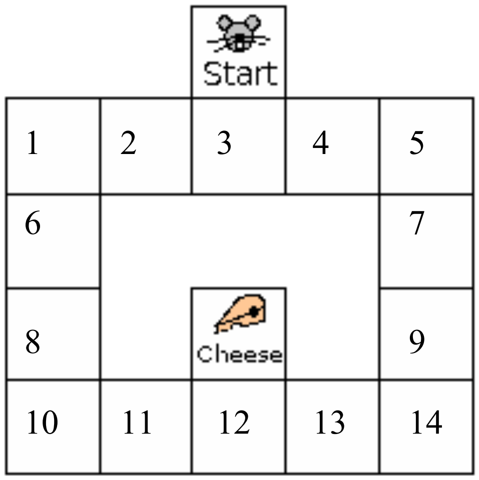
(a) For the game above, assuming the operators have unit cost, consider the following possible heuristics. For each, say whether it is admissible (yes or no) and justify your answer.

i. $h(n) = 0$
ii. $h(n) = \#$ pieces on cells in the goal zone (lower right corner).
iii. $h(n) = \sum_i h_i(n)$, where $h_i(n)$ = Manhattan distance between a piece i's current position to the closest cell in the goal zone.
iv. $h(n) = \max_i \{h_i\}$, where h_i is defined as above.

(b) Now consider the case where moves to an adjacent cell are half the cost of jumping over another piece. For each of the heuristics in (a), say whether it is admissible (yes or no) and justify your answer.

Question 3: Search under uncertainty

Consider the following problem (from *http://users.isr.ist.utl.pt/~mtjspaan/readingGroup/index_en.html*):



An intelligent mouse must move about in the maze so as to get the cheese as quickly as possible. At each square, it must choose whether to move up, down, left or right. There is some uncertainty in the outcome of the actions of the mouse. For example, it may happen that, sometimes, the action "Move left" the mouse two squares to the left instead of just one. On the other hand, the mouse must try to realize in which square of the maze it is by observing its surroundings. Of course, there are many squares that look alike, and so the mouse isn't always able to tell exactly where it is...

We assume there are 16 states, corresponding to the grid cells shown above, {Start,1,2,3, …, 14, Cheese}. The goal state is the cell showing the cheese. There are 4 move actions {Up, Down, Left Right}. Each can cause the mouse to either: move over by 1 cell in the intended direction, or move over by 2 cells in the intended directions (if the move is blocked by a wall, the mouse stays in place). The mouse knows the layout of the maze, but cannot observe its position fully. Instead, in each state, the mouse can only perceive the number of adjacent walls, {1, 2, 3, 4}, but not their orientation. *E.g. In state {1}, the mouse observes {2} since there are 2 adjacent walls.*

(a) What is the initial belief state?

(b) What is the total number of possible beliefs?

(c) Consider the case where the initial belief is limited to the top row: {1, 2, 3, 4, 5}. What is the belief state after the action sequence {Down, Left, Left, Down} (assuming no observations)?

(d) For the same case of an initial belief over the top row, give a conformant plan that allows the mouse to reach the cheese using no observations.

(e) For the same case of an initial belief over the top row, draw the first three levels of the AND-OR search tree.

Question 4: Constraint satisfaction (Adapted from Poole&Mackworth.)

A classic example of a constraint satisfaction problem is a crossword puzzle. This example contains six three-letter words: three words across (labeled A1, A2, A3) and three words down (labeled D1, D3, D3). Each word must be chosen from the list of forty possible words shown on the right.

A1,D1	D2	D3
A2		
A3		

add, ado, age, ago, aid, ail, aim, air, and, any, ape, apt, arc, are, ark, arm, art, ash, ask, auk, awe, awl, aye, bad, bag, ban, bat, bee, boa, ear, eel, eft, far, fat, fit, lee, oaf, rat, tar, tie.

(a) There are many alternate representations in terms of variables, including:
  i. A representation where the domains of the variables correspond to words in English.
  ii. A representation where the domains of the variables correspond to letters of the alphabet.

For each of these representations, describe the set of variables, give the number of variables that would be needed to capture the specific puzzle shown above.

(b) Using the first of these representations, draw the constraint graph for this problem.

(c) Apply unary constraints on this graph and give the (possibly) reduced domain of each variable.

(d) Apply arc-consistency in this graph, and give the (possibly) reduced domain of each variable.

(e) Apply backtracking search, with the relevant heuristics (minimum-remaining-values, degree, least-constraining-value) to solve this problem. Break ties using alphabetical ordering. Give your final solution.

Question 5: Iterative improvement algorithms

(a) Explain how to formulate the CSP problem above so that it can be solved by hill-climbing. Explain what is a state (or node) for this representation, the set of operators, and the objective function.

(b) Write code that does the following: (1) randomly generates a large number of initial solutions for this problem, by filling the grid cells with random letters, (2) for each initial solution, runs hill climbing to find a correct solution. Measure the search cost and percentage of solved problems, and graph these against the optimal solution cost. Comment on your results. This should be reported in the pdf portion of your homework.

Also submit the code used to produce these results. Your code should read-in the input file in the format shown here: *http://www.cs.mcgill.ca/~jpineau/comp424/Homework/example_input.txt*

(This is very important! We will be testing your code using a different version of the CSP.)

Sample code (in python) is provided here:

http://www.cs.mcgill.ca/~jpineau/comp424/Homework/csp_base_code.txt (change the extension to .py)

This file can be run from the command line in either of two modes.

- To run a simple demo, which reads from "input_file" and writes a set of states (in the format which the grading script can understand) to "output_file", type: "python 1 input_file output_file".

To run a demo of the grading process, which also serves to verify that the output of your code will be parseable by the grading script, given the file input_file which was input to your code and the file output_file which was output by your code, type: "python 2 input_file output_file".

Note: Basically, you want the "score" associated your input/output file pair to be as _large_ as possible, corresponding to a greater reduction in conflicts from the start state in input_file to the least conflicted state described in output_file.

Note: For students coding in java, your code should be executable as follows: "java csp_solver.java input_file output_file", where input_file will be in the same format as the example input file, and output_file is such that running the current file in "grading" mode (i.e. mode 2) gives a reasonable result. If your submitted file is not named csp_solver.java, you will receive no credit.