CS472 Foundations of Artificial Intelligence
Fall 2000
Assignment 1

Due Monday, September 11 at the beginning of class.

Collaboration: You are allowed to work in groups of 2-4 students for this assignment. Collaboration between groups is not allowed. Working alone is fine. In all cases, you must write up the solutions yourself; no collaboration is allowed for the write-up.

1. **Uninformed Search (10 pts.)**

   (a) (5 pts.) Construct a finite search tree for which it is possible that depth-first search uses *more* memory than breadth-first search. (Be sure to show the goal node(s) in your tree.) Is there any tree and distribution of goal nodes for which depth-first search *always* requires more memory than breadth-first search? Briefly justify your answer.

   (b) (5 pts.) Give a modification to the depth-first search algorithm that would reduce the space requirements to $O(d)$, rather than $O(bd)$.

2. **Formulating Search Problems (20 pts.)**

   (a) (15 pts.) Specify each of the following problems as a search problem. Note that there are several possible formulations for each problem, with varying levels of detail. The main thing is that your formulations should be precise and consistent enough so that they could be implemented/simulated.

   For example, consider the 8-squares problem. We will represent states as a 3-tuple of 3-tuples (each representing the contents of a row). Thus, the final state – where the numbered tiles are placed in order around the edge of the board – would be represented as ((1 2 3)(8 B 4)(7 6 5)) where B is the blank. There would be four operators, one to move a tile each of North, South, East and West. For example, the West-move operator would only be legal if the B were in the second or third position in a tuple; its effect would be to swap the B with the number immediately to its left. Path costs for the 8-squares problem would all be the same, with the cost of a solution equal to the number of steps necessary to solve the puzzle.

      i. (5 pts.) You want to find the telephone number of Turing-award winner (and Cornell CS professor) Juris Hartmanis, who lives in Aurora, given a stack of phone directories alphabetically ordered by city. (Note that Prof. Hartmanis doesn't actually live in Aurora, in case you're trying to find him!)

      ii. (5 pts.) Same as in (a), but you have forgotten Juris's last name.

      iii. (5 pts.) You want to *generate* a crossword puzzle, given a crossword grid and a dictionary of legal words. (See question 4 for an example of a simple crossword puzzle grid.)

   (b) (5 pts.) Give an example of a problem that *can't* be expressed as a search problem. Justify your answer.

3. **Iterative Deepening (10 pts.)**

   (a) (5 pts.) Is iterative deepening a good idea for the crossword puzzle generation problem described above? Why or why not?

   (b) (5 pts.) Consider the following algorithm for *iterative broadening*:

      i. Set the breadth cutoff, c, to 2.

      ii. Set L to be the set of initial node(s), i.e. nodes associated with the initial state(s).

      iii. Let n be the first node on L. If L is empty, increment c and return to step ii.

      iv. If n is a goal node, stop and return it (and the path from the initial node to n).

      v. Otherwise, remove n from L. Add to the front of L the first c of n's children (labeling each with its path from the initial node). Return to step iii.

   Would you expect iterative broadening to be any better on the crossword puzzle generation problem? Why or why not?

4. **Constraint Satisfaction Problems (25 pts.)**

   A technique called *arc-consistency* can be used to reduce the size of the search space for large CSP problems. A *constraint network* is a representation of a CSP where the variables of a CSP are nodes and each binary constraint is a bidirectional arc between the two variables involved in the constraint. (Constraints must be binary.) An arc $< X, Y >$ is *arc-consistent* if for each value of X in the domain of X (D_X) there is some value Y in the domain of Y (D_Y) such that the constraint between X and Y is satisfied. If an arc $< X, Y >$ is *not* arc-consistent, all values of X in D_X for which there is no corresponding value in D_Y may be deleted to make $< X, Y >$ arc-consistent – thus reducing the size of the CSP and making search more effective. The entire constraint network can be made arc-consistent by repeatedly considering potentially inconsistent arcs until all are shown to be consistent. (Note that if we remove values from some domain D_X, all arcs $< Z, X >$ where $Z \neq Y$ become potentially inconsistent, because some value of X may have been removed which had previously been necessary for $< Z, X >$ to be arc-consistent.)

   Try the CSP applet at http://www.cs.ubc.ca/labs/lci/CIspace/ – the sample "Scheduling Problem" should help you understand how arc-consistency works. (It's much easier than it sounds!)

   For unary constraints, such as $X \neq 5$, a simpler technique called "domain consistency" simply removes all the domain values that don't satisfy the unary constraint. (It's possible to avoid unary constraints altogether by specifying the domains of the variables to only include the values that would have satisfied the unary constraints – this is why the CSP applet doesn't support unary constraints.)

   Consider the crossword puzzle given in Figure 1. Suppose we have the following words in our dictionary: {ant, ape, big, bus, car, has, bard, book, buys, hold, lane, year, rank, browns, ginger, symbol, syntax}. The goal is to fill the puzzle with words from the dictionary.

   (a) (10 pts.) Formalize the problem as a constraint satisfaction problem. (There are several ways to do this, but choosing the most natural way will probably make the following questions easier. You're welcome to use unary constraints in the CSP specification, if you find them useful.)

   (b) (5 pts.) The resulting constraint network is not arc-consistent. Give one domain value that can be pruned. Explain which constraint arc can be used to prune it, and why.

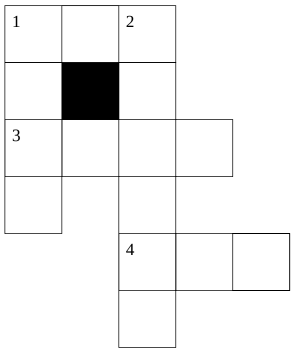

Figure 1: A simple crossword puzzle.

(c) (10 pts.) Give the domains after arc-consistency stops. (You may wish to use the CIspace CSP applet to make this task easier.)

5. **Heuristic Search (15 pts.)**

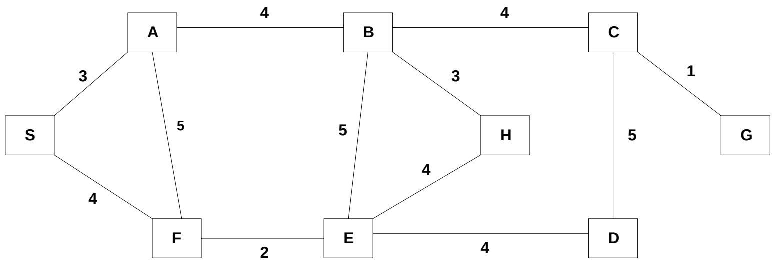

Figure 2: Graph for Heuristic Search question.

See the graph in Figure 2. The start state is S and the goal state is G. The numbers on the arcs indicate the cost of traversing that arc. Nodes should be expanded alphabetically should the heuristic used have the same evaluation priority for two or more nodes. You may, to reduce the number of nodes expanded, use a cycle-checking variant of the search algorithms which never considers neighbors of a node which are already on the path from the start – see the second bullet item in section 3.6 of R & N for more details.

Whenever the search algorithm requires a heuristic estimating function, use the following:

n	S	A	B	C	D	E	F	G	H
h(n)	8	7	4	2	3	5	6	0	1

You may want to try using the CIspace graph-searching applet at http://www.cs.ubc.ca/labs/lci/CIspace/ to help automate the searching.

(a) (5 pts.) Using the graph in Fig. 2 and **uniform-cost search** (a.k.a. **lowest-cost-first search**):

i. list the nodes in the order they would be expanded

ii. list the nodes that lie along the final correct path to the goal.

(b) (5 pts.) Same as (a), but using **greedy best-first search**.

(c) (5 pts.) Same as (a), but using **A* search**.

What to turn in: You'll have to turn in BOTH a paper and an electronic copy of the assignment.

- **On paper:** All assignments for this course must be typewritten. (Those portions of any assignment that are easier to draw by hand can be drawn in by hand.) For this assignment, turn in the answers to each question separately. (We'll have five boxes in which to deposit your assignment: one box for solutions to question 1, one box for solutions to question 2, etc.) At the top of the first page of the solution for each question, include (1) your name, (2) your e-mail address, and (3) an alphabetically ordered list of everyone in your homework group.
- **Electronic:** To protect you in the unlikely case that your homework is lost after you turn it in, we also require that you submit an electronic copy of your work (by the date and time shown above).
  1. Create a single ZIP file that contains all of the files connected with your homework solution. If you have only one file, please still create a ZIP file for it anyway (because the default transfer mode for the ftp server is binary). Unix users can use *tar* and *gzip*.
  2. Name your file with your e-mail address followed by an underscore and the assignment number, in this case "1". The final filename extension will be either ZIP or *gz* depending on what you did in step 1. So aaa999@cornell.edu_1.ZIP or aaa999@cornell.edu_1.tar.gz would be two possible filenames for this assignment.
  3. Next, FTP the file to "playpen.cs.cornell.edu" with username "cs472" and password "deepblue". Note that this directory provides write-only, no-overwrite access. This means that you get to upload exactly one version of your assignment.

  Please see the course web site for more details on electronic submission and for instructions for what to do if you need to submit a revised version of the assignment.