CSE 471

Homework 1

For the following questions, please keep your answers as brief as possible. No reason to fill full pages.

Part A:

1. We talked about accessible/inaccessible, static/dynamic and deterministic/non-deterministic classifications for an environment. Comment, as precisely as you can, on which of these classifications depend also on the agent's own capabilities. For example, can the same environment be accessible for one agent and inaccessible (or less accessible) for another? Deterministic for one agent and non-deterministic (less deterministic) for another? Static for one agent and less static (dynamic) for another agent? What properties/capabilities of the agent wind up being critical in changing the classification?
2. Use the vacuum agent scenario we discussed in the book to illustrate examples to justify your answers below.
    a. Consider an environment E that is accessible for the agent A1 and inaccessible for the agent A2. Answer the following:
        i. Will A2 be able to achieve ***any*** goals in E that A1 can achieve?
        ii. Will A2 be able to achieve ***all*** goals that A1 can achieve?
        iii. For the goals that both A1 and A2 can achieve, will A2 be able to attain the same level of performance that A2 can achieves (a) in terms of the time it takes to figure out what actions it should do (b) "quality" of those actions
    b. Do the question 2(a) above but with respect to environment E' that is deterministic for A1 and stochastic for A2.
3. It can be shown that asymptotically, Iterative deepening Depth first search expands (b+1)/(b-1) nodes more than depth first search, when given a uniform tree of depth d and branching factor b. For the case where b=1, this formula evaluates to infinity.

Does this make sense? If not, can you compute the ratio for this special case?

4. We have a uniform search tree of depth d and branching factor b, where all the solutions are in the depth d, but the solutions are distributed *NON-UNIFORMLY*--they may all be clustered together.
    a. If you have to choose between using depth first or breadth first search, which would you choose?
    b. Consider the following blind search strategy that works in iterations:

    Loop for b'= 1 to b
    do
    Generate a "sub-tree" S(b') where, for each node in the original tree, only the first b' of its children are considered (the rest are ignored in this iteration).

    Search S(b') for a goal
    If a goal is found, get out of the loop and return it
    od
    (Assume that the search of S(b') is done using the algorithm you chose in 4(a)).

        i. Argue that this search is likely to do better on the average than the best search you picked in 4(a).
        ii. Draw a tree for b=3 and d=3 and show the subtrees searched in each iteration.
        iii. In many problems, if you make the wrong first move, you will then go into a region of the search tree that is completely barren of goal nodes. Argue that the algorithm in 4(b) effectively improves your chances, on the average, of doing well on such problems.
5. Suppose we have a heuristic h that over-estimates h* by at most epsilon (i.e., for all n, 0<= h(n) <= h*(n)+epsilon). Show that A* search using h will get a goal whose

cost is guaranteed to be at most epsilon more than that of the optimal goal.

6. Consider the following variant of the A* algorithm that we call New-A* algorithm. New-A* algorithm uses a queue management strategy that is different from A*:
   If there is a goal node on the open and it has the lowest f-value among all nodes on open, return it and terminate. If not, select a non-goal node with the largest f-value from open list and expand it, adding the children to the open list. Suppose the heuristic used of new-A* is admissible.
   a. Is new-a* guaranteed to terminate on all goal-bearing trees?
   b. If new-a* returns a solution, is it guaranteed to be optimal?
   c. Is new-a* going to be more or less efficient in general than A*?

   Justify your answers.
7. Consider a version of A* where we use as evaluation function f(n) = (2 - w) g(n) + w h(n) where w is between 0 and 2; and h is an admissible heuristic.
   Consider the case where w=2. What sort of search is this? Is it guaranteed to be optimal? Will it be optimal if h=h*? How does it compare to hill-climbing search with a goodness function h(n)?

Part B:

One of my favorite exam questions is of the following type. So, you might as well practice doing them. The following are from a previous year's midterm.
*For each of the following statements below, indicate whether the statement is true or false, and give a brief but precise justification for your answer. Correct answers *with correct justifications* will carry 2 points. No points will be awarded for answers without correct justifications.*

***Example Question**: The time and memory requirements of IDA* can be improved by using A* algorithm to do search in individual iterations.*

Answer: *False. Because A* in the worst case can take as much memory as breadth-first, and thus using A* in the individual iterations will make IDA* require exponential memory (instead of linear memory).*

A. A* search does b^(d/2) node expansions when searching a uniform tree of branching factor b and depth d, using a perfect heuristic.
B. Consider a uniform search tree of depth d and branching factor b, where there are many goal nodes, all of which are uniformly distributed at the leaf level d. Assuming that memory consumption is not a problem, we are better off using breadth-first search than depth-first search in this scenario.
C. A* search with heuristic h=0 will always have to search the entire tree before finding the optimal solution.
D. Suppose A* search uses an evaluation function f(n) = (1-w) g(n) + w h(n). For any value of w between 0 and 1 (inclusive), A* will terminate and return the optimal solution.
E. If h1 and h2 are two admissible heuristics, and h3 is defined as h3(n) = max(h1(n) , h2(n)) , then A* search with h3 is guaranteed to return an optimal solution while expanding as many or fewer nodes than either h1 or h2.
F. For finite search trees, Hill climbing is a complete search strategy (i.e., will always terminate), but may sometimes return inoptimal solutions.

Part C:

I. Consider the 8-puzzle problem where the goal-state is ("0" is the blank tile):

1	***2***	***3***
4	***0***	***5***
6	***7***	***8***

Suppose we choose the top row [1,2,3] as the fringe pattern.

Consider the following initial state-call it S0

3	***4***	***5***
1	***0***	***2***
7	***6***	***8***

1. What is the heuristic value associated with S0 if we use
    a. Misplaced tile heuristic
    b. Manhattan distance heuristic
    c. Pattern Database heuristic-assuming top row as the fringe pattern.

    You will have to compute the value for 1(c.) - since the puzzle is small enough you should be able to do it by hand (kind of good exercise playing the sliding puzzle games):
2. Compare the heuristic values. How close is the PDB value to the perfect heuristic value in this case?
3. Assuming that we use top row as the fringe pattern, how many distinct entries will you need in the pattern database? What fraction of the total number of states in 8-puzzle is this database size?