0	1	2	3	4	5	6	total
0	*20*	*25*	*30*	*30*	*25*	*20*	*150*

Name: ____________________________________

UMBC 671 Midterm Exam

19 October 2009

Write all of your answers on this exam, which is closed book and consists of six problems, summing to 160 points. You have the entire class period, seventy five minutes, to work on this exam. Good luck.

0. Warm Up Exercise (0)

There are three labeled boxes containing fruit. One is labeled "apples", another "oranges" and the third "mixed". You know that each is labeled incorrectly. You may ask for a piece of fruit from one box, which you choose. Is there a way to correctly label the boxes? (Hint: Save this one for last)

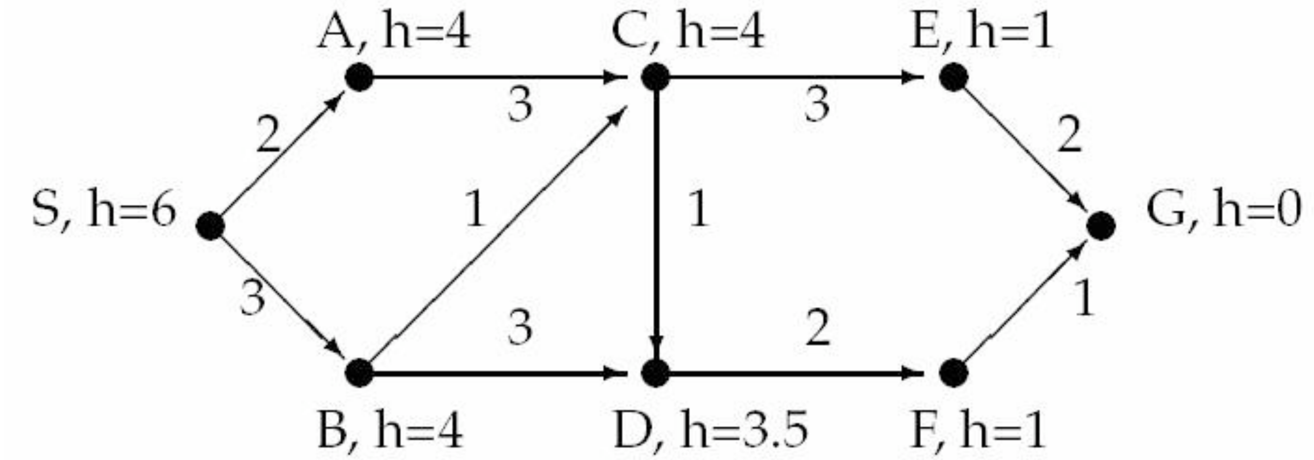
1. True/False [20 points]

Circle either T or an F in the space before each statement to indicate whether the statement is true or false. If you think the answer is simultaneously true and false, quit while you are ahead. There is no penalty for incorrect answers but then, there are no points for incorrect answers either.

T F A stochastic environment is one in which the next state is completely determined by the agent’s action.
T F A nominally fully observable environment can be rendered partially observable due to errors and noise in an agent's sensors.
T F Depth-first search always expands at least as many nodes as A* search with an admissible heuristic.
T F The major drawback of Hillclimbing is that it is only guaranteed to work for two-dimensional search spaces.
T F It is possible for Algorithm A* to expand a non-optimal solution node when using an admissible heuristic.
T F Algorithm A will perform a breadth-first search if F(n) = G(n).
T F The main advantage of the Iterative Deepening A* algorithm over the standard A* algorithm is that its memory requirement is only linear in the depth of the search.
T F A simple breadth-first search will always find an optimal solution if one exists and it is of finite length and all nodes have a finite number of successors.
T F Iterative Deepening search is a practical way to add heuristics to algorithm A.
T F Algorithm A will perform a depth-first search if f(n) = - g(n).
T F An advantage of Hill Climbing search is that it requires minimal memory.
T F Bi-directional search is always more efficient than uni-directional search.
T F Arc consistency is a more powerful constraint propagation algorithm than forward checking.
T F Deciding if a CSP is consistent is, in general, NP-hard.
T F The alpha-beta algorithm is preferred to minimax because it provides a better estimation of which move is best for a given look-ahead distance.
T F A drawback of game playing strategies using the minimax and alpha-beta algorithms is that one can’t make sacrifices where an immediate loss is permitted to get a subsequent advantage.
T F The minimax and alpha-beta procedures will always back up identical values to the root node of a game tree for any possible static evaluation function.
T F A perfectly rational backgammon agent never loses.
T F The prisoner’s dilemma shows that game theory is not sound.
T F Game theory applies only to zero-sum games.

2. Search I [25]

The graph below represents the search space of a problem. Nodes are labeled with a letter and the value of a heuristic function h for the node. Edges are labeled with the cost of traversing the edge. Suppose we want to use the A* algorithm on the graph to find the shortest path from node S to node G.



(a) [15] Simulate running the A* algorithm on the graph, filling in lines in the table, showing the f, g, and h values of each node on the queue. We've done the first two lines as an example. If you find a new path to a node already on the queue, update its cost (using the lower f value) instead of adding another copy of that node to the queue. Keep the queue sorted.

(b) [10] For the solution found by A*, give the cost and sequence of nodes comprising the path.

Iteration	*Node expanded*	*Priority Queue (aka fringe, open list) Notation for a node is NAME(F value, G value, H value)*
0	--	S(6,0,6)
1	S	A(6,2,4), B(7,3,4)
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		

3. Linear tile puzzle [30]

Consider a sliding tile puzzle with six tiles (three black, three white) in a linear tray holding seven tiles. The following is the initial configuration **[B][B][B][W][W][W][E]** where [B] is a black tile, [W] a white one and [E] the 'empty' cell. The puzzle has these moves: (1) A tile may move into an adjacent empty cell with unit cost; and (2) A tile may hop over one or two tiles into the empty cell with cost equal to the number of tiles hopped over. Thus, the initial configuration has the following successors:

[B][B][B][W][W][E][W] (cost = 1)
[B][B][B][W][E][W][W] (cost = 1)
[B][B][B][E][W][W][W] (cost = 2)

The goal is to have all of the white tiles to the left of all of the black ones. It is unimportant where the empty cell is. Your task is to set this problem up for solution by algorithm A.

(a) [3] Give a loose upper bound on the size of the search space in terms of the number of states by ignoring the fact that identity of the three tiles of each color is unimportant.

(b) [3] Give a tighter upper bound on the number of states that does reflect the fact that the tiles of a given color are interchangeable.

(c) [4] What is the maximum number of successors that a state can have? What's the minimum? Estimate the average number of successors.

(d) [4] How many states are goal states

(e) [4] Is every action in this puzzle reversible?

(f) [4] Suggest a reasonable representation for this puzzle in Python or Java so that it can be used with the AIMA searching package.

(g) [4] Describe a heuristic function (h) appropriate for this problem.

(h) [4] Is your heuristic estimate admissible? Justify your answer in a sentence or two

4. Constraint Satisfaction Problems [30 points]

There are five MWF graduate CS classes and three instructors who will be teaching these classes. An instructor can only teach one class at a time. The classes are:

- Class 1 - Algorithms: meets from 8:00-9:00am
- Class 2 - Intro to AI: meets from 8:30-9:30am
- Class 3 - Databases: meets from 9:00-10:00am
- Class 4 - Operating Systems: meets from 9:00-10:00am
- Class 5 - Machine Learning: meets from 9:30-10:30am

and the instructors are:

- Professor A, who is available to teach Classes 3 and 4.
- Professor B, who is available to teach Classes 2, 3, 4, and 5.
- Professor C, who is available to teach Classes 1, 2, 3, 4, 5.

(a) [8] Formulate this as a CSP problem in which there is one variable per class. State the variables and their domains, and the constraints. Constraints should be unary or binary and specified formally and precisely, but may be implicit rather than explicit.

(b) [7] Draw the constraint graph associated with your CSP.

(c) [8] Show the domains of the variables after running arc-consistency on this initial graph (after having already enforced any unary constraints).

(d) [7] Give one solution to this CSP.

5. NimChop [25]

NimChop is a two-person game played with a pile of stones. To play this game, at each move, the player must divide one of the remaining piles of stones into two nonempty piles of different sizes. Thus, six stones may be divided into piles of 5/1, 4/2 but not 3/3. The player who can no longer make a move loses the game.

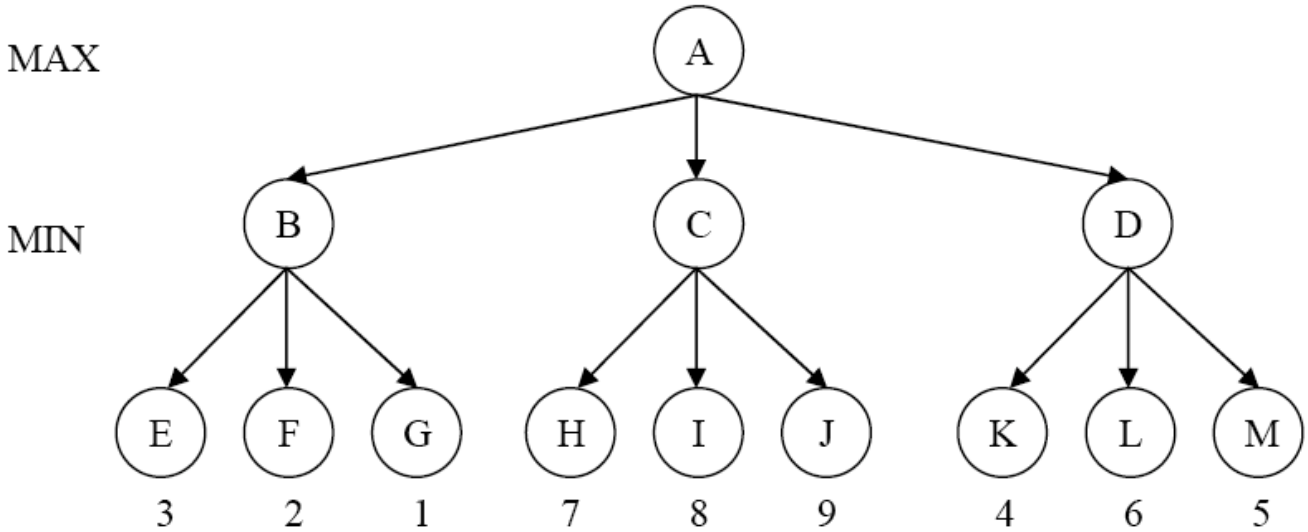
(a) [5] Is this a zero sum game? Is this a game where each player has perfect information?

(b) [15] Draw the complete game tree or graph when the initial pile has seven stones and using it, determine what the outcome is (e.g., win for player one, win for player two, draw, unknown) if both players play optimally.

(c) [5] What is the outcome if the initial pile has nine stones?

6. Game Trees [20]

Consider this game tree where the root is a maximizing node, and children are visited left to right.



(a) [5] Compute the minimax game value of nodes A, B, C, and D using the standard algorithm.

A = ; B = ; C = ; D =

(b) [5] What move will be selected by player one using minimax.

(c) [5] List the nodes (leaves or interior) that alpha-beta prunes, i.e., decides need not be examined.

(c) [5] The number of nodes that alpha-beta prunes depends on how each node's children are ordered in the tree. Draw a new game tree by re-ordering the children of each internal node, such that the new game tree is equivalent to the tree above, but alpha-beta pruning will prune as many nodes as possible. List the nodes that would be pruned.