CS 188 Spring 2012 Introduction to Artificial Intelligence Practice Midterm

To earn the extra credit, one of the following has to hold true. Please circle and sign.

A I spent 2 or more hours on the practice midterm.

B I spent fewer than 2 hours on the practice midterm, but I believe I have solved all the questions.

Signature: _____

The normal instructions for the midterm follow on the next page.

- You have 2 hours.
- The exam is closed book, closed notes except a two-page crib sheet.
- Please use non-programmable calculators only.
- Mark your answers ON THE EXAM ITSELF. If you are not sure of your answer you may wish to provide a *brief* explanation. All short answer sections can be successfully answered in a few sentences AT MOST.

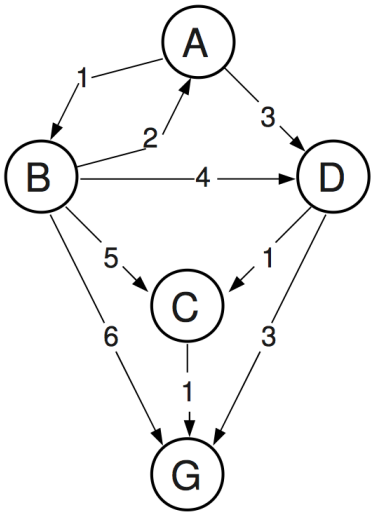
First name	
Last name	
SID	
Login	

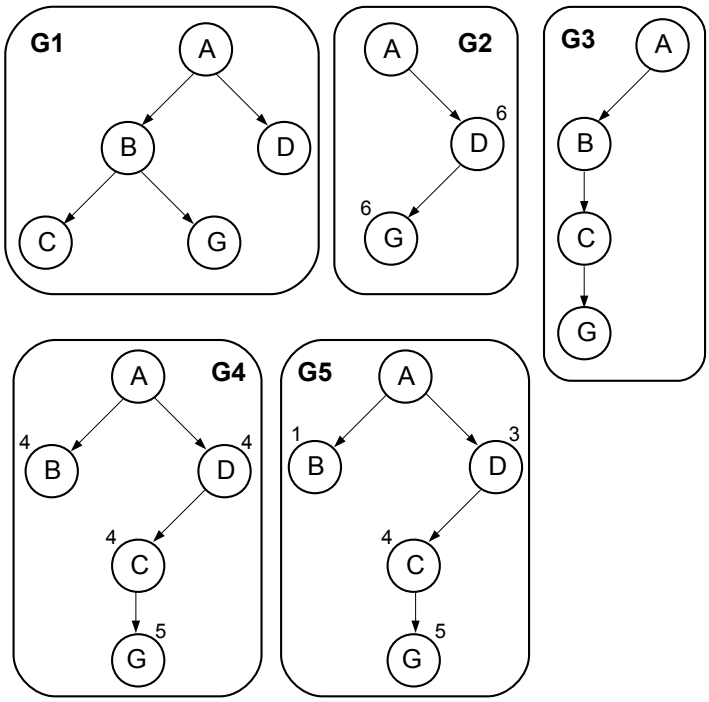
For staff use only:

Q1. Search Traces	/15
Q2. Minimax and Expectimax	/12
Q3. n-pacmen search	/10
Q4. Crossword Puzzles as CSPs	/13
Q5. Short answer	/17
Total	/67

Q1. [15 pts] Search Traces

Each of the trees (G1 through G5) was generated by searching the graph (below, left) with a *graph search* algorithm. Assume children of a node are visited in alphabetical order. Each tree shows *only the nodes that have been expanded*. Numbers next to nodes indicate the relevant "score" used by the algorithm's priority queue. The start state is A, and the goal state is G. Note: we follow the protocol in our lecture that goal check only takes place during the expansion of a node, NOT when we add a node to the fringe.





For each tree, indicate:

1. Whether it was generated with depth first search, breadth first search, uniform cost search, or A^* search. Algorithms may appear more than once.
2. *If* the algorithm uses a heuristic function, say whether we used

$$\boldsymbol{H}_1 = \{h(A) = 3, h(B) = 6, h(C) = 4, h(D) = 3\}$$
$$\boldsymbol{H}_2 = \{h(A) = 3, h(B) = 3, h(C) = 0, h(D) = 1\}$$

3. For all algorithms, say whether the result was an optimal path (assuming we want to minimize sum of link costs). If the result was *not* optimal, state why the algorithm found a suboptimal path.

Please fill in your answers on the next page.

(a) [3 pts] **G1**:

1. Algorithm:
2. Heuristic (if any):
3. Did it find least-cost path? If not, why?

(b) [3 pts] **G2**:

1. Algorithm:
2. Heuristic (if any):
3. Did it find least-cost path? If not, why?

(c) [3 pts] **G3**:

1. Algorithm:
2. Heuristic (if any):
3. Did it find least-cost path? If not, why?

(d) [3 pts] **G4**:

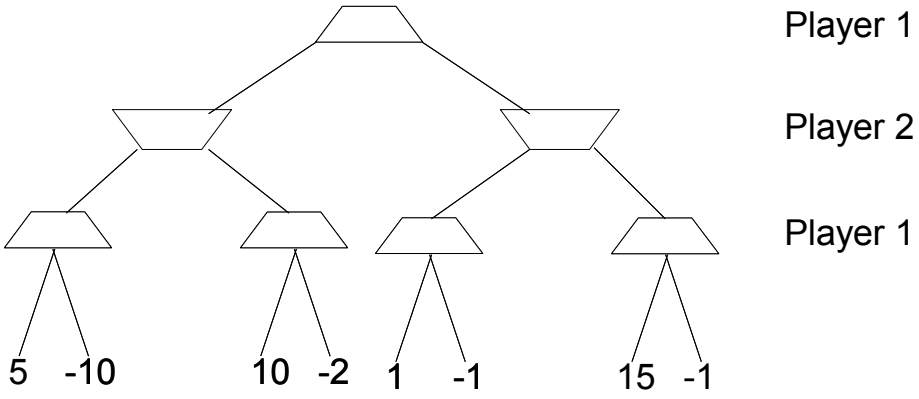
1. Algorithm:
2. Heuristic (if any):
3. Did it find least-cost path? If not, why?

(e) [3 pts] **G5**:

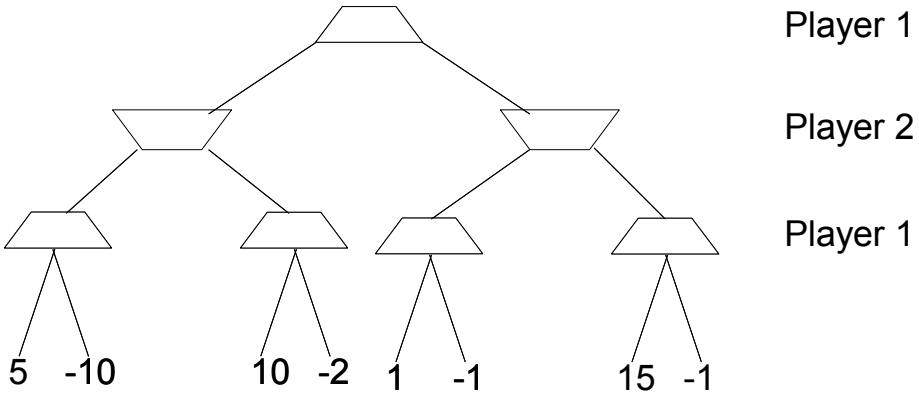
1. Algorithm:
2. Heuristic (if any):
3. Did it find least-cost path? If not, why?

Q2. [12 pts] Minimax and Expectimax

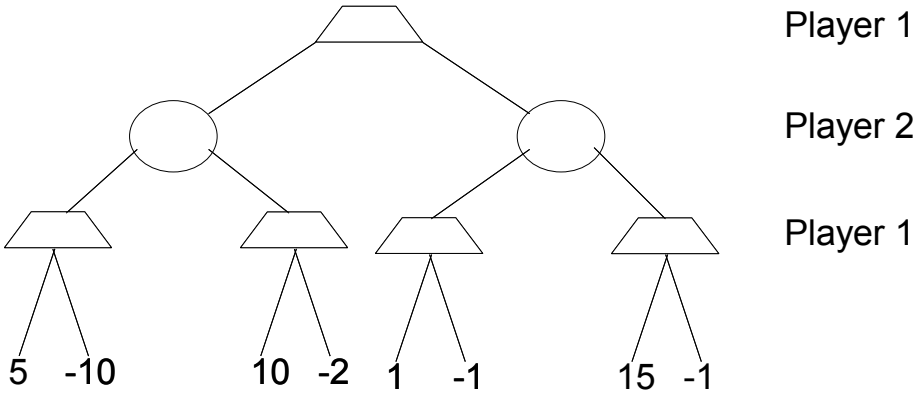
(a) [2 pts] Consider the following zero-sum game with 2 players. At each leaf we have labeled the payoffs Player 1 receives. It is Player 1's turn to move. Assume both players play optimally at every time step (i.e. Player 1 seeks to maximize the payoff, while Player 2 seeks to minimize the payoff). Circle Player 1's optimal next move on the graph, and state the minimax value of the game. Show your work.



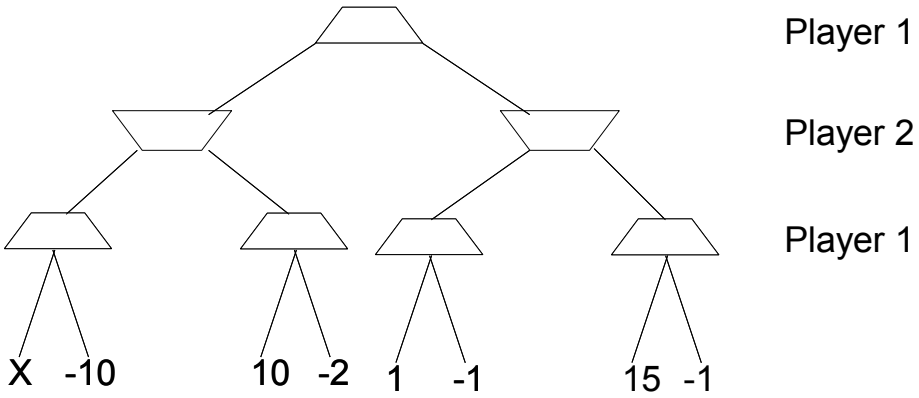
(b) [2 pts] Consider the following game tree. Player 1 moves first, and attempts to maximize the expected payoff. Player 2 moves second, and attempts to minimize the expected payoff. Expand nodes left to right. Cross out nodes pruned by alpha-beta pruning.



(c) [2 pts] Now assume that Player 2 chooses an action uniformly at random every turn (and Player 1 knows this). Player 1 still seeks to maximize her payoff. Circle Player 1's optimal next move, and give her expected payoff. Show your work.



Consider the following modified game tree, where one of the leaves has an unknown payoff x. Player 1 moves first, and attempts to maximize the value of the game.



(d) [2 pts] Assume Player 2 is a minimizing agent (and Player 1 knows this). For what values of x does Player 1 choose the left action?

(e) [2 pts] Assume Player 2 chooses actions at random (and Player 1 knows this). For what values of x does Player 1 choose the left action?

(f) [2 pts] For what values of x is the minimax value of the tree worth more than the expectimax value of the tree?

Q3. [10 pts] n-pacmen search

Consider the problem of controlling n pacmen simultaneously. Several pacmen can be in the same square at the same time, and at each time step, each pacman moves by at most one unit vertically or horizontally (in other words, a pacman can stop, and also several pacmen can move simultaneously). The goal of the game is to have all the pacmen be at the same square in the minimum number of time steps. In this question, use the following notation: let M denote the number of squares in the maze that are not walls (i.e. the number of squares where pacmen can go); n the number of pacmen; and $p_i = (x_i, y_i) : i = 1 \dots n$, the position of pacman i. Assume that the maze is connected.

(a) [1 pt] What is the state space of this problem?

(b) [1 pt] What is the size of the state space (not a bound, the exact size).

(c) [2 pts] Give the tightest upper bound on the branching factor of this problem.

(d) [2 pts] Bound the number of nodes expanded by uniform cost *tree* search on this problem, as a function of n and M. Justify your answer.

(e) [4 pts] Which of the following heuristics are admissible? Which one(s), if any, are consistent? Circle the corresponding Roman numerals and briefly justify all your answers.

1. The number of (ordered) pairs (i, j) of pacmen with different coordinates: $h_1(s) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} (p_i \neq p_j)$.
   (i) Consistent? (ii) Admissible?

2. $h_2(s) = \frac{1}{2} \max(\max_{i,j} |x_i - x_j|, \max_{i,j} |y_i - y_j|)$.
   (i) Consistent? (ii) Admissible?

Q4. [13 pts] Crossword Puzzles as CSPs

You are developing a program to automatically solve crossword puzzles, because you think a good income source for you might be to submit them to the New York Times ($200 for a weekday puzzle, $1000 for a Sunday).[1] For those unfamiliar with crossword puzzles, a crossword puzzle is a game in which one is given a grid of squares that must be filled in with intersecting words going from left to right and top to bottom. There are a given set of starting positions for words (in the grid below, the positions $1, 2, 3, 4$, and 5), where words must be placed going across (left to right) or down (top to bottom). At any position where words intersect, the letters in the intersecting words must match. Further, no two words in the puzzle can be identical. An example is the grid below, in which the down words (1, 2, and 3) are `DEN`, `ARE`, and `MAT`, while the across words (1, 4, and 5) are `DAM`, `ERA`, and `NET`.

Example Crossword Grid and Solution

1D	2A	3M
4E	R	A
5N	E	T

A part of your plan to make crosswords, you decide you will create a program that uses the CSP solving techniques you have learned in CS 188, since you want to make yourself obsolete at your own job from the get-go. Your first task is to choose the representation of your problem. You start with a dictionary of all the words you could put in the crossword puzzle, where the dictionary is of size K and consists of the words $\{d_1, d_2, \ldots, d_K\}$. Assume that you are given a grid with N empty squares and M different entries for words (and there are 26 letters in the English language). In the example above, $N = 9$ and $M = 6$ (three words across and three words down).

You initially decide to use words as the variables in your CSP. Let D_1 denote the first down word, D_2 the second, D_3 the third, etc., and similarly let A_k denote the kth across word. For example, in the crossword above, $A_1 =$ `DAM`, $D_1 =$ `DEN`, $D_2 =$ `ARE`, and so on. Let $D_1[i]$ denote the letter in the ith position of the word D_1.

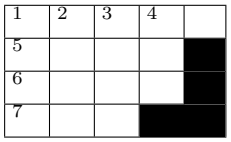
(a) [1 pt] What is the size of the state space for this CSP?

(b) [3 pts] Precisely (i.e. use mathematical notation to) describe the constraints of the CSP when we use words as variables.

[1] `http://www.nytimes.com/2009/07/19/business/media/19askthetimes.html`

After defining your CSP, you decide to go ahead and make a small crossword using the grid below. Assume that you use the words on the right as your dictionary.

Crossword Grid



Dictionary Words

ARCS, BLAM, BEAR, BLOGS, LARD, LARP,
GAME, GAMUT, GRAMS, GPS, MDS, ORCS, WARBLER

(c) [1 pt] Enforce all *unary* constraints by crossing out values in the table below.

D_1	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
D_2	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
D_3	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
D_4	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
A_1	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
A_5	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
A_6	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
A_7	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER

Here's an extra table in case you make a mistake:

D_1	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
D_2	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
D_3	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
D_4	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
A_1	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
A_5	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
A_6	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
A_7	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER

(d) [1 pt] Assume that in backtracking search, we assign A_1 to be GRAMS. Enforce unary constraints, and in addition, cross out all the values eliminated by forward checking against A_1 as a result of this assignment.

D_1	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
D_2	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
D_3	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
D_4	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
A_1	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
A_5	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
A_6	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
A_7	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER

Here's an extra table in case you make a mistake:

D_1	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
D_2	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
D_3	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
D_4	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
A_1	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
A_5	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
A_6	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
A_7	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER

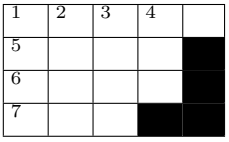
(e) [3 pts] Now let's consider how much arc consistency can prune the domains for this problem, even when no assignments have been made yet. I.e., assume no variables have been assigned yet, enforce unary constraints first, and then enforce arc consistency by crossing out values in the table below.

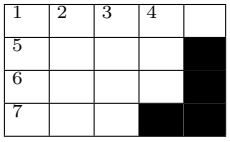
D_1	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
D_2	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
D_3	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
D_4	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
A_1	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
A_5	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
A_6	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
A_7	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER

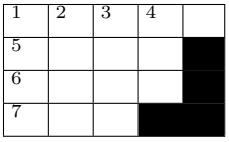
Here's an extra table in case you make a mistake:

D_1	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
D_2	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
D_3	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
D_4	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
A_1	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
A_5	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
A_6	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
A_7	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER

(f) [1 pt] How many solutions to the crossword puzzle are there? Fill them (or the single solution if there is only one) in below.







Your friend suggests using letters as variables instead of words, thinking that sabotaging you will be funny. Starting from the top-left corner and going left-to-right then top-to-bottom, let X_1 be the first letter, X_2 be the second, X_3 the third, etc. In the very first example, $X_1 = \texttt{D}$, $X_2 = \texttt{A}$, and so on.

(g) [1 pt] What is the size of the state space for this formulation of the CSP?

(h) [2 pts] Assume that in your implementation of backtracking search, you use the least constraining value heuristic. Assume that X_1 is the first variable you choose to instantiate. For the crossword puzzle used in parts (c)-(f), what letter(s) might your search assign to X_1?

Q5. [17 pts] Short answer

Each true/false question is worth 1 point. Leaving a question blank is worth 0 points. **Answering incorrectly is worth -1 point.**

(a) For a search problem, the path returned by uniform cost search may change if we

(i) [*true* or *false*] rescale all step costs by a scalar α: $0 < \alpha < 1$.

(ii) [*true* or *false*] rescale all step costs by a scalar α: $1 < \alpha < 2$.

(iii) [*true* or *false*] add a positive constant C to every step cost.

(b) Assume we are running A^* graph search with a consistent heuristic h. Let p be the node in the fringe about to be expanded in the search. When expanding p, we find that exactly one of its children is the goal state G and the cost of the found path through p to G is K. Let pathcost(p) denote the cost of the path to p that led to p being inserted in the queue. Then we have that

(i) [*true* or *false*] the found path through p to the goal is a shortest path.

(ii) [*true* or *false*] the found path through p to the goal is guaranteed to be at most $K - \text{pathcost}(p)$ longer than the shortest path.

(iii) [*true* or *false*] the found path through p to the goal is guaranteed to be at most $K - \text{pathcost}(p) - h(p)$ longer than the shortest path.

(iv) [*true* or *false*] the found path through p to the goal is guaranteed to be the shortest path going through p to the goal state.

(c) Consider two consistent heuristics, H_1 and H_2, in an A^* search seeking to minimize path costs in a graph. Assume ties do not occur in the priority queue. If $H_1(s) \leq H_2(s)$ for all s, then

(i) [*true* or *false*] A^* search using H_1 will find a lower cost path than A^* search using H_2.

(ii) [*true* or *false*] A^* search using H_2 will find a lower cost path than A^* search using H_1.

(iii) [*true* or *false*] A^* search using H_1 will not expand more nodes than A^* search using H_2.

(iv) [*true* or *false*] A^* search using H_2 will not expand more nodes than A^* search using H_1.

(d) Consider the following statements about α-β pruning. Alpha-beta pruning

(i) [*true* or *false*] may not find the minimax optimal strategy.

(ii) [*true* or *false*] prunes the same number of subtrees independent of the order in which successor states are expanded.

(iii) [*true* or *false*] generally requires more run-time than minimax on the same game tree.

(e) Consider a zero-sum game adversarial game. The minimizer is played by a computer program that is fast enough to perform min-max search all the way to the end of the game and it plays according to the thus-found moves. It is the minimizer's turn to play, and the minimizer's computer program returns a win of some positive value for the minimizer. Then we have that

(i) [*true* or *false*] the minimizer is guaranteed to win the game only if the maximizer also plays the min-max strategy.

(ii) [*true* or *false*] the minimizer is guaranteed to win the game only if the maximizer plays a deterministic strategy.

(iii) [*true* or *false*] if the maximizer is known to make moves uniformly at random every other turn, then the minimizer is not necessarily maximizing pay-off.