# CS221 Practice Midterm

Autumn 2012

## 1 Other Midterms

The following pages are excerpts from similar classes' midterms. The content is similar to what we've been covering this quarter, so that it should be useful for practicing. Note that the topics and terminology differ slightly, so feel free to ignore the questions that we did not cover (e.g., greedy search, Bayes nets). The real midterm will lie somewhere between these problems and the homework problems in style and difficulty.

**3. (15 points.) CSPs**

You are in charge of scheduling for computer science classes that meet Mondays, Wednesdays and Fridays. There are 6 classes that meet on these days and 3 professors who will be teaching these classes. You are constrained by the fact that each professor can only teach one class at a time.

The classes are:

- Class 1 - Intro to Programming: meets from 8:00-9:00am
- Class 2 - Intro to Artificial Intelligence: meets from 8:30-9:30am
- Class 3 - Natural Language Processing: meets from 9:00-10:00am
- Class 4 - Computer Vision: meets from 9:00-10:00am
- Class 5 - Machine Learning: meets from 10:30-11:30am

The professors are:

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

**(1) (4 pts):** Formulate this problem as a CSP problem in which there is one variable per class, stating the domains, and constraints. Constraints should be specified formally and precisely, but may be implicit rather than explicit.

**(2) (2 pts):** Draw the constraint graph associated with your CSP.

**(4) (4 pts):** Show the domains of the variables after running arc-consistency on this initial graph (after having already enforced any unary constraints).

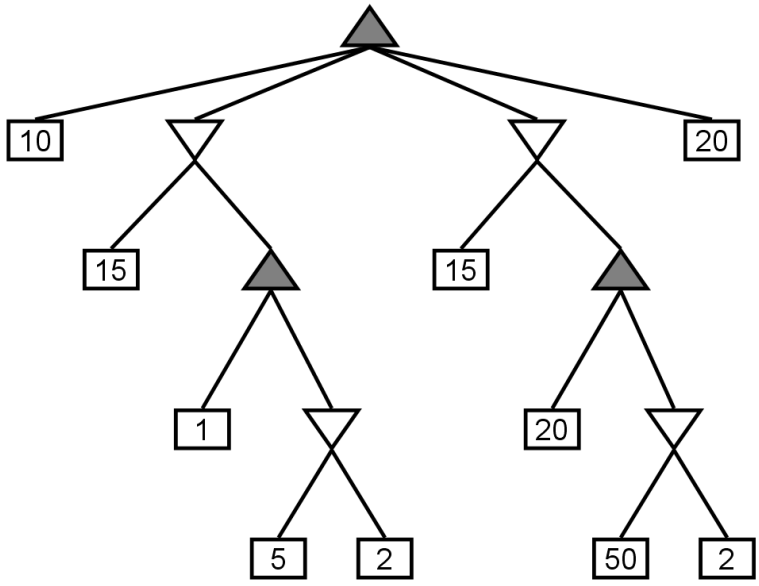
**(5) (1 pt):** Give one solution to this CSP.

**(6) (2 pts):** Your CSP should look nearly tree-structured. Briefly explain (one sentence or less) why we might prefer to solve tree-structures CSPs.

**(7) (2 pts):** Name (or briefly describe) a standard technique for turning these kinds of nearly tree-structured problems into tree-structured ones.

**4. (12 points.) Minimax Search**

Consider the following minimax tree.




**(1) (2 pts):** What is the minimax value for the root?

**(2) (5 pts):** Draw an X through any nodes which will not be visited by alpha-beta pruning, assuming children are visited in left-to-right order.

**(3) (2 pts):** Is there another ordering for the *children of the root* for which more pruning would result? If so, state the order.

**(4) (3 pts):** Propose a general, practical method for ordering children of nodes which will tend to increase the opportunities for pruning. You should be concise, but clearly state both what to do about min nodes and max nodes.

**1. (12 points.) Search: Mr. and Ms. Pacman**

Pacman and Ms. Pacman are lost in an $N$x$N$ maze and would like to meet; *they don't care where*. In each time step, *both* simultaneously move in one of the following directions: {NORTH, SOUTH, EAST, WEST, STOP}. They do *not* alternate turns. You must devise a plan which positions them together, somewhere, in as few time steps as possible. Passing each other does not count as meeting; they must occupy the same square at the same time.

**(a) (4 points)** Formally state this problem as a *single-agent* state-space search problem.

**States:**

**Maximum size of state space:**

**Maximum branching factor:**

**Goal test:**

**(b) (3 points)** Give a non-trivial admissible heuristic for this problem.

**(c) (3 points)** Circle all of the following graph search methods which are guaranteed to output optimal solutions to *this problem*:

(i) DFS
(ii) BFS
(iii) UCS
(iv) A* (with a consistent and admissible heuristic)
(v) A* (with heuristic that returns zero for each state)
(vi) Greedy search (with a consistent and admissible heuristic)

**(d) (2 points)** If $h_1$ and $h_2$ are admissible, which of the following are also guaranteed to be admissible? Circle all that apply:

(i) $h_1 + h_2$
(ii) $h_1 * h_2$
(iii) $max(h_1, h_2)$
(iv) $min(h_1, h_2)$
(v) $(\alpha)h_1 + (1 - \alpha)h_2$, for $\alpha \in [0, 1]$

**2. (11 points.) CSPs: Finicky Feast**

You are designing a menu for a special event. There are several choices, each represented as a variable: (A)ppetizer, (B)everage, main (C)ourse, and (D)essert. The domains of the variables are as follows:

A: (v)eggies, (e)scargot
B: (w)ater, (s)oda, (m)ilk
C: (f)ish, (b)eef, (p)asta
D: (a)pple pie, (i)ce cream, (ch)eese

Because all of your guests get the same menu, it must obey the following dietary constraints:

(i) Vegetarian options: The appetizer must be veggies or the main course must be pasta or fish (or both).

(ii) Total budget: If you serve the escargot, you cannot afford any beverage other than water.

(iii) Calcium requirement: You must serve at least one of milk, ice cream, or cheese.

**(a) (3 points)** Draw the constraint graph over the variables A, B, C, and D.

**(b) (2 points)** Imagine we first assign A=e. Cross out eliminated values to show the domains of the variables after forward checking.

| | | | | |
|---|---|---|---|---|
| A | [ | e | | ] |
| B | [ | w s m | | ] |
| C | [ | f b p | | ] |
| D | [ | a i ch | | ] |

**(c) (3 points)** Again imagine we first assign A=e. Cross out eliminated values to show the domains of the variables after arc consistency has been enforced.

| | | | | |
|---|---|---|---|---|
| A | [ | e | | ] |
| B | [ | w s m | | ] |
| C | [ | f b p | | ] |
| D | [ | a i ch | | ] |

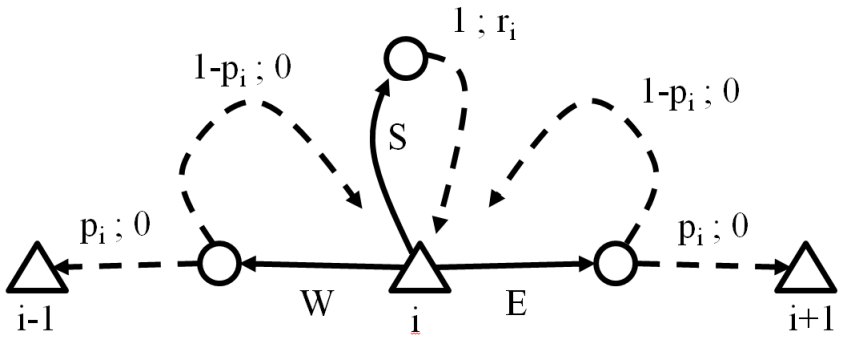
**(d) (1 point)** Give a solution for this CSP or state that none exists.

**(e) (2 points)** For general CSPs, will enforcing arc consistency after an assignment *always* prune at least as many domain values as forward checking? Briefly explain why or why not.

**4. (17 points.) MDPs and RL: Wandering Merchant**

There are $N$ cities along a major highway numbered 1 through $N$. You are a merchant from city 1 (that's where you start). Each day, you can either travel to a neighboring city (actions *East* or *West*) or stay and do business in the current city (action *Stay*). If you choose to travel from city $i$, you successfully reach the next city with probability $p_i$, but there is probability $1 - p_i$ that you hit a storm, in which case you waste the day and do not go anywhere. If you stay to do business in city $i$, you get $r_i > 0$ in reward; a travel day has reward 0 regardless of whether or not you succeed in changing cities.

The diagram below shows the actions and transitions from city $i$. Solid arrows are actions; dashed arrows are resulting transitions labeled with their probability and reward, in that order.



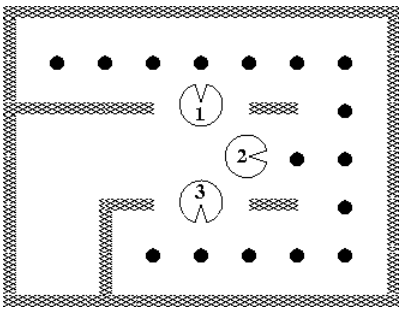

**(a) (2 points)** If for all $i$, $r_i = 1$, $p_i = 1$, and there is a discount $\gamma = 0.5$, what is the value $V^{stay}(1)$ of being in city 1 under the policy that always chooses *stay*? Your answer should be a real number.

**(b) (2 points)** If for all $i$, $r_i = 1$, $p_i = 1$, and there is a discount $\gamma = 0.5$, what is the optimal value $V^*(1)$ of being in city 1?

**(c) (2 points)** If the $r_i$'s and $p_i$'s are known positive numbers and there is almost no discount, i.e. $\gamma \approx 1$, describe the optimal policy. You may define it formally or in words, e.g. "always go east," but your answer should precisely define how an agent should act in any given state. *Hint:* You should not need to do any computation to answer this question.

**2. (10 points) Cooperative Pac-Family**

Pacman is trying eat all the dots, but he now has the help of his family! There are initially $k$ dots, at positions $(f_1, \ldots f_k)$. There are also $n$ Pac-People, at positions $(p_1, \ldots, p_n)$; initially, all the Pac-People start in the bottom left corner of the maze. Consider a search problem in which all Pac-People move *simultaneously*; that is, in each step each Pac-Person moves into some adjacent position (N, S, E, or W, no STOP). Note that any number of Pac-People may occupy the same position.




**(a) (3 pt)** Define the state space of the search problem.

**(b) (1 pt)** Give a reasonable upper bound on the size of the state space for a general $r$ by $c$ grid.

**(c) (1 pt)** What is the goal test?

**(d) (1 pt)** What is the maximum branching factor of the successor function in a general grid?

**(e) (4 pt)** Circle the admissible heuristics below (-1/2 point for each mistake.)

i. $h_1(s) = 0$
ii. $h_2(s) = 1$
iii. $h_3(s) =$ number of remaining food / $n$
iv. $h_4(s) = \max_i \max_j \text{manhattan}(p_i, \text{food}_j)$
v. $h_5(s) = \max_i \min_j \text{manhattan}(p_i, \text{food}_j)$
vi. $h_6(s) = \min_i \max_j \text{manhattan}(p_i, \text{food}_j)$
vii. $h_7(s) = \min_i \min_j \text{manhattan}(p_i, \text{food}_j)$
viii. $h_8(s) = \max(h_3, h_7)$
ix. $h_9(s) = \min(h_3, h_4)$
x. $h_{10}(s) =$ Solve the single-agent food search problem for each Pac-Person individually from their current position. Return the minimum of these divided by $n$.

**1. (16 points) True/False**

For the following questions, a correct answer is worth 2 points, no answer is worth 1 point, and an incorrect answer is worth 0 points. Circle *true* or *false* to indicate your answer.

a) (*true* or *false*) If $g(s)$ and $h(s)$ are two admissible A$^*$ heuristics, then their average $f(s) = \frac{1}{2}g(s) + \frac{1}{2}h(s)$ must also be admissible.

b) (*true* or *false*) For a search problem, the path returned by uniform cost search may change if we add a positive constant $C$ to every step cost.

c) (*true* or *false*) The running-time of an efficient solver for tree-structured constraint satisfaction problems is linear in the number of variables.

d) (*true* or *false*) If $h_1(s)$ is a consistent heuristic and $h_2(s)$ is an admissible heuristic, then $\min(h_1(s), h_2(s))$ must be consistent.

e) (*true* or *false*) The amount of memory required to run minimax with alpha-beta pruning is $O(b^d)$ for branching factor $b$ and depth limit $d$.

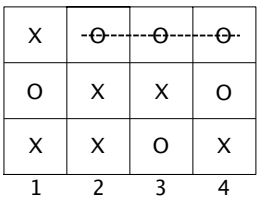
f) (*true* or *false*) In a Markov decision process with discount $\gamma = 1$, the difference in values for two adjacent states is bounded by the reward between them: $|V(s) - V(s')| \leq \max_a R(s, a, s')$.

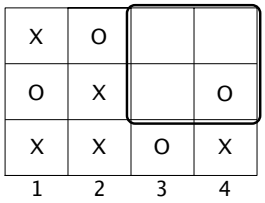
g) (*true* or *false*) Value iteration and policy iteration must always converge to the same policy.

h) (*true* or *false*) In a Bayes' net, if $A \perp\!\!\!\perp B$, then $A \perp\!\!\!\perp B \mid C$ for some variable $C$ other than $A$ or $B$.

**4. (10 points) Multi-agent Search: Connect-3**

In the game of Connect-3, players X and O alternate moves, dropping their symbols into columns 1, 2, 3, or 4. Three-in-a-row wins the game, horizontally, vertically or diagonally. X plays first.



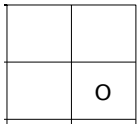




**(a) (1 pt)** What is the maximum branching factor of minimax search for Connect-3?

**(b) (1 pt)** What is the maximum tree depth in plies?

**(c) (1 pt)** Give a reasonably tight upper bound on the number of terminal states.

**(d) (2 pt)** Draw the game tree starting from the board shown above right (with O to play next). You may abbreviate states by drawing only the upper-right region circled. The root node is drawn for you.




**(e) (2 pt)** X is the maximizer, while O is the minimizer. X's utility for terminal states is $k$ when X wins and $-k$ when O wins, where $k$ is 1 for a horizontal 3-in-a-row win (as in above left), 2 for a vertical win, and 3 for a diagonal win. A tie has value 0. Label each node of your tree with its minimax value.

**(f) (3 pt)** Circle all nodes of your tree that will *not* be explored when using alpha-beta pruning and a move ordering that maximizes the number of nodes pruned.