UNIVERSITY OF PENNSYLVANIA

CIS 391/521: Fundamentals of AI
Midterm 1, Spring 2010

Question	Points
1 Environments	/2
2 Python	/18
3 Local and Heuristic Search	/35
4 Adversarial Search	/20
5 Constraint Satisfaction	/25
Total	/100

Name: ____________________________________

Penn Email:____________________________________

Exam policy: This exam is closed-book; no printed, hand-written, laptop-available, or cell phoned material can be consulted during the exam.

Time: 80 minutes.

1 Environments [2 points]

1. [1 points] TRUE or FALSE: Driving a car offers a dynamic environment.

   ★ **SOLUTION:** True. Other cars, pedestrians, etc. are part of the environment and are constantly moving.

2. [1 points] TRUE or FALSE: Playing chess, checkers, or go offers a partially observable environment. (In case you're not familiar with one of the games, the answer is the same for each game.)

   ★ **SOLUTION:** False. The entire board is visible to all players.

2 Python [18 points]

2.1 Multiple choice [12 points]

Circle **all** responses that satisfy the statements below; there may be several items you should circle for a single question. If none of the responses satisfy the statement, circle "none of the above".

1. [2 points] Indicate which statements create a **list** with the elements 1, 2, and 3.

   (a) $[1, 2, 3]$
   (b) range(1, 3)
   (c) $(1, 2, 3)$
   (d) [i for i in 1, 2, 3]
   (e) none of the above

   ★ **SOLUTION:** a, d

2. [2 points] Indicate which types could be dictionary **keys**.

   (a) integer
   (b) string
   (c) tuple
   (d) list
   (e) set
   (f) none of the above

★ **SOLUTION:** a, b, c

3. [2 points] Indicate which types could be dictionary **values**.

   (a) integer
   (b) string
   (c) tuple
   (d) list
   (e) set
   (f) none of the above

★ **SOLUTION:** all

★ **SOLUTION:** b, c

4. [2 points] Suppose you have a class called SudokuBoard which contains the function printBoard (just like in HW1). Indicate which of the following statements should be used to call printBoard from **within** the class.

   (a) printBoard()
   (b) self.printBoard()
   (c) printBoard(self)
   (d) self.printBoard(self)
   (e) none of the above

★ **SOLUTION:** b

5. [2 points] Python is dynamically typed. This means:

   (a) The data type of a particular variable cannot change from one block of code to the next
   (b) The data types of variables are determined automatically
   (c) The language has some primitive types and some object types
   (d) Python programmers tend to be energetic in their keystrokes

★ **SOLUTION:** b. And d.

2.2 Short answer [6 points]

1. [3 points] For each block of code, on the line below it write what would be printed by the “print x” statement.

Block A	Block B	Block C
x = [1, 2]	x = 1	x = “python”
y = x	y = x	y = x
y[1] = 3	y = 3	y = y.upper()
print x	print x	print x
______________	______________	______________

★ **SOLUTION:** [1, 3], 1, python

2. [3 points] Write the result of this code: print [[j for i in 1, 2] for j in 3, 4]

★ **SOLUTION:** [[3, 3], [4, 4]]

3 Local and Heuristic Search [35 points]

3.1 Multiple Choice [9 points]

For each of the following questions, circle the **single correct answer**.

1. [2 points] Which of the following is true?

   a. All admissible heuristics are consistent
   b. All consistent heuristics are admissible
   c. All of the above
   d. None of the above

   ★ **SOLUTION:** b. Consistency is a stronger property than admissibility.

2. [3 points] What is true if A* is run with a heuristic that is not admissible? Assume that the objective is to find the least-cost path to a goal state.

   a. The solution is not guaranteed to be optimal
   b. The solution is guaranteed to be suboptimal
   c. A* may never return a solution, given an infinite search space
   d. (a), (b), and (c)
   e. (a) and (c)
   f. The space-time continuum will collapse and the universe will end

   ★ **SOLUTION:** e

3. [2 points] Assuming a *maximization* problem, what is the primary role of "temperature" in Simulated Annealing?

   a. Adjusts the probability of taking a "good" (uphill) move
   b. Adjusts the probability of taking a "bad" (downhill) move
   c. Controls how much memory the algorithm uses
   d. Sets the algorithm's initial state
   e. Helps students stay warm during the winter

★ **SOLUTION:** b

4. [2 points] Assuming a *maximization* problem, if the "temperature" stays too high, Simulated Annealing can potentially:

   a. Take too many downhill steps and never reach a local or global maxima

   b. Take too few uphill steps and get stuck in a local maxima

   c. Temperature has no practical impact on the convergence of simulated annealing

   d. Run of out memory

   e. Give the user 3rd degree burns

★ **SOLUTION:** a

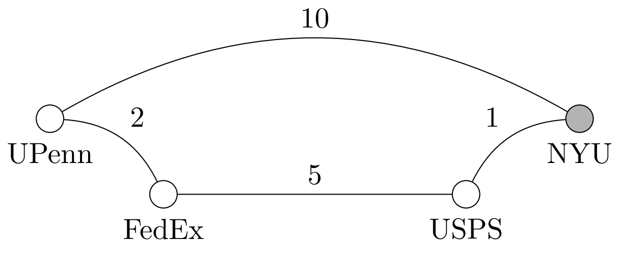
3.2 Short Answer [26 points]

1. [6 points] Given branching factor b, maximum search tree depth m, and least cost solution depth d, write down the (worst-case) time and space complexity using Big-O notation for the following:

	Time	Space
Depth-First Search (DFS):		
Breadth-First Search (BFS):		
Iterative Deepening Search (IDS):		

★ **SOLUTION:** DFS: Time: $O(b^m)$, Space: $O(bm)$; BFS: Time: $O(b^{d+1})$, Space: $O(b^{d+1})$; IDS: Time: $O(b^d)$, Space: $O(bd)$.

2. Consider the graph below, which represents two possible ways of mailing packages from the University of Pennsylvania to NYU: by carrier pigeon (UPenn →NYU) or by commercial carriers (UPenn →FedEx →USPS →NYU).



   (a) [2 points] What is the cost of the optimal path from UPenn to NYU?

★ **SOLUTION:** The cost of path UPenn →FedEx →USPS →NYU is 8.

(b) [6 points] Consider the following heuristics, h_1, h_2 and h_3, estimating the cost from the specified node to the goal node, NYU.

	UPenn	FedEx	USPS	NYU
h_1	10	1	1	0
h_2	5	1	1	0
h_3	5	4	1	0

For each of the heuristics, circle whether it is admissible and/or consistent. (Circle one, both, or neither for each heuristic.) You do NOT need to explain your answer.

i. h_1: Admissible Consistent

ii. h_2: Admissible Consistent

iii. h_3: Admissible Consistent

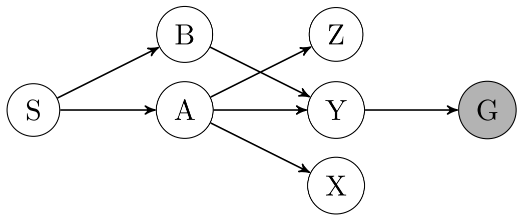
★ **SOLUTION:** h_2 admissible, h_3 admissible and consistent. h_1 is not admissible (and therefore not consistent) because h_1(UPenn) is greater than the optimal path. h_2 underestimates and is admissible, but it is not consistent because h_2(UPenn) $> h_2$(FedEx) $+ c$(UPenn →FedEx); i.e., f(UPenn) = 5, but f(FedEx) = 3, so f is *not* non-decreasing. For h_3, f along optimal path is $\{0+5=5, 2+4=6, 7+1=8, 8+0=8\}$, so h_3 is consistent and therefore admissible.

(c) [3 points] How many times will A* expand FedEx if h_2 is used as the heuristic, if A* does **NOT** keep track of visited states?

★ **SOLUTION:** Once.

■ **COMMON MISTAKE:** f(UPenn →FedEx) = 2+1 = 3, and f(UPenn →FedEx →UPenn →FedEx) = 2+1+2+2 = 7, which are both < 8 and *would* be visited **if** h_2 **were consistent.** However, h_2 is *not* consistent: f(UPenn →FedEx →Upenn) = 2+2+5 = 9, which means that the optimal path will be discovered before UPenn is expanded for a second time. Therefore FedEx will be visited only once.

3. [6 points] Write down the sequence of states (NOT the fringe) considered by DFS and BFS in the following **directed** graph, starting from state S, and assuming alphabetical ordering of states (i.e., when there is an arbitrary choice of which success state to exand, pick the one that is first in the alphabet) and that the search stops once the algorithm reaches the goal state G.



i. DFS:

ii. BFS:

★ **SOLUTION:** DFS: S, A, X, Y, G. BFS: S, A, B, X, Y, Z, G *or* S,A,B,X,Y,Z,Y,G (depending on whether repeated states were allowed.)

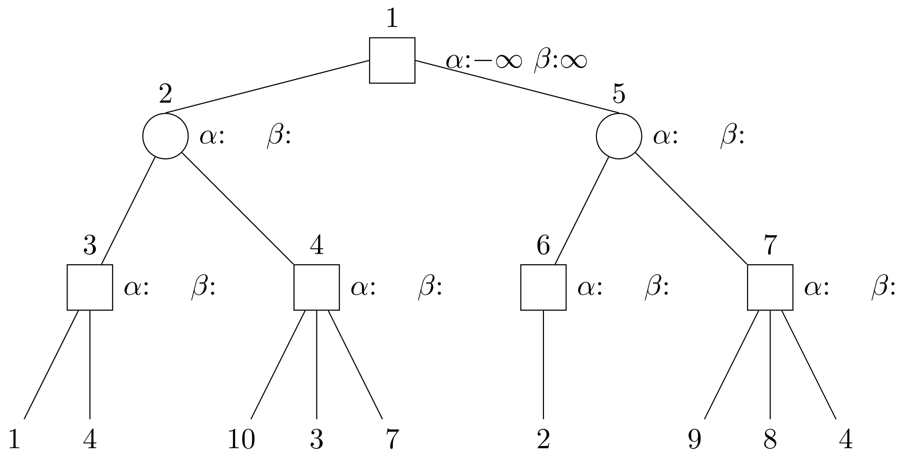
4. [3 points] You own a company that manufactures goods A, B, and C, that sell for \$5, \$2 and \$1 per item respectively. 100, 60 and 30 grams of raw material apiece are required for A, B, and C, respectively. You have 1000 grams of material available. Cast the problem of **maximizing profit** as a linear programming problem, using a, b, and c to represent the numbers of each good being made. (Note: You just need to write the problem in LP form, not solve it.)

★ **SOLUTION:** $\max 5a + 2b + c$, subject to $100a + 60b + 30c \leq 1000$ and $a, b, c \geq 0$.

■ **COMMON MISTAKE:** Forgetting the lower bound constraints ensuring that a, b, and c are non-negative. Without this, consider the solution $c = -4x$, $b = 0, a = 10 + x$: then $100(10 + x) - 30(4x) = 1000 - 20x$, so the constraint is always satisfied. However, profit is then $5(10 + x) - 4x = 50 + x$. So we see that we can choose x to be arbitrarily large and get arbitrary profit.

4 Adversarial Search [20 points]

The following tree represents all possible outcomes of a hypothetical zero-sum game:



This tree is from the perspective of the MAX player; **MAX nodes are repesented by squares and MIN nodes by circles**. The leaves of the tree represent the value of the game for the MAX player. The number of each node indicates the order in which they are considered by the Minimax and α-β pruning algorithms.

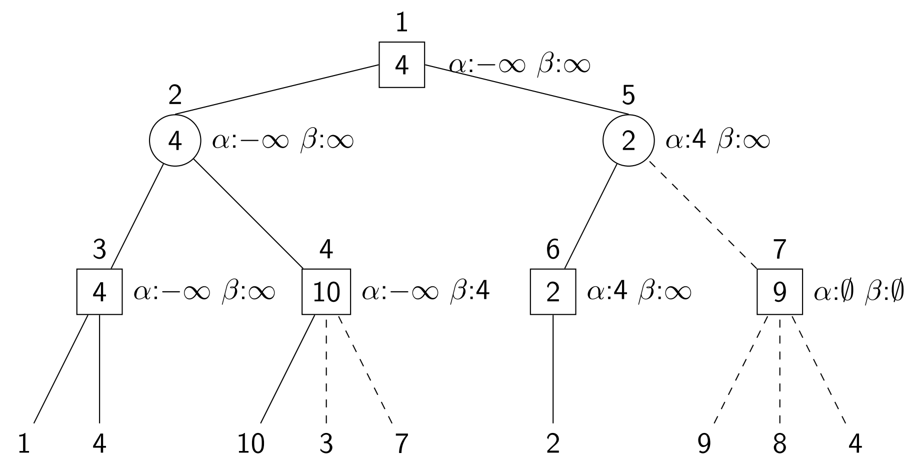
1. [8 points] Compute the "back-up" values of each node in the tree using the Minimiax strategy, and write these values into the space inside each node.

2. *Note: this question has two parts; read both completely before starting to answer the question.*

   i. [6 points] Run the α-β pruning algorithm and circle each leaf **and** node that would NOT be considered by the α-β pruning algorithm. (Assume that leaves are considered in left-to-right order.)

   ii. [6 points] Furthermore, as you do so, write the values of α and β that are **passed as arguments** to the recursive call at each node in the space provided. (For example, the α-β pruning algorithm is initialized with $\alpha = -\infty$ and $\beta = \infty$, so these are the values passed into the first call at the root node.) Note: If any nodes don't receive α or β values due to pruning, just write "x" for their α and β values.

★ **SOLUTION:** In this picture dashed lines indicate pruned edges.



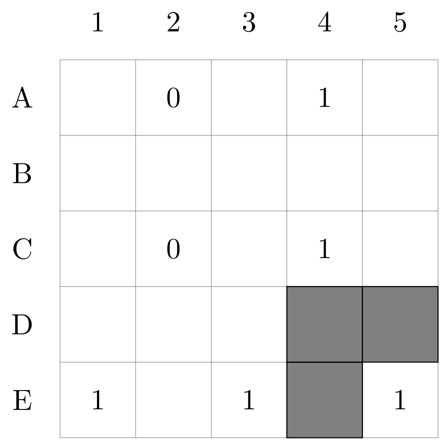
■ **COMMON MISTAKE:** 1 point was taken off for each leaf or node that was not either incorrectly not circled or incorrectly circled. A common mistake was to only circle nodes and not **leaves**.

■ **COMMON MISTAKE:** 1/2 point was taken off for each incorrect α or β value, or if the right (α, β) values were present but swapped. (Swapping counted as a single 1/2 point mistake). The key point in this problem was to remember that α and β are computed from backed-up values but that α and β are only passed *down*. So, by definition, the backed-up value chosen by a min or max node can affect the α and β of its parent node, but only for subsequent calls on other siblings from the same parent. So, node 3 returns 4 and thus $\beta = 4$ for node 4, but this β value is *not* passed back up to node 2. Instead, node 2 returns 4 back to node 1, and thus node 1 sets $\alpha = 4$ for the evaluations at node 5 and subsequent nodes.

The other key point was to remember that the task was to write the α and β that were *input* to the α-β pruning algorithm at each node.

5 Creepy Constraint Satisfaction [25 points]

You have a friend Stan who is a police investigator. He has received a letter from an inside source at a chemical company, Chemically Really Especially Evil Products (CREEPs). The letter contained a coded message telling Stan that CREEPs plans to dump toxic residuals from their factory in **3 locations** around the city tonight. The letter didn't say where the locations are exactly (the source was afraid to pinpoint them, in case the letter should fall into the hands of his supervisors), but it did contain the map you see below.



	1	2	3	4	5
A		0		1	
B					
C		0		1	
D				▨	▨
E	1		1	▨	1

Figure 1: Each grid square represents one location in the city.

Stan thinks **a number in a map square indicates how many dumping locations are adjacent to that square**. For example, the 1 in the lower right corner of the map would mean that one of the three locations adjacent to this square are dumping locations. (We've shaded those locations on the map to be clear about which ones we're talking about here.) Assuming Stan is correct, and also that **the numbered squares are not themselves dumping locations**, let's consider this from the perspective of constraint satisfaction.

The variables for this problem are the 25 squares. Call them A1, A2, etc. (based on the letters and numbers on the outside edge of the grid). Before doing any inference, all variables have the same domain: +1 for "is a dumping location" and 0 for "is not a dumping location".

1. [2 points] In terms of the variables (A1, A2, . . ., E5), write an equation for the global constraint imposed by the fact that there are exactly 3 dumping locations.

   ★ **SOLUTION:** A1 + ... + E5 = 3

   ■ **COMMON MISTAKE:** Some people said this sum should be 8, because they added in 5 extra locations for the "1"s on the board. However, the squares marked as "1" on the board are **not** dumping locations. (This is specified in the problem setup.)

2. The 1 in E5 can be seen as imposing 1 node constraint and 3 arc constraints:

- [1 points] The node constraint is on E5. Write this constraint as an equation in terms of the value of E5 (as in "E5 = ...").

  ★ **SOLUTION:** E5 $= 0$

  ■ **COMMON MISTAKE:** Some people said E5 $= 1$, since there is a 1 in square E5 on the board. However, the node constraint is on the **value of the variable** E5. Since numbered squares are not dumping locations, the value of E5 is 0.

- [4 points] The 3 arc constraints are on the variables D4, D5, and E4. Write the 3 equations describing these constraints. *(Note that E5 also imposes the constraint D4 + D5 + E4 = 1, but this is not an arc constraint since it involves 3 variables.)*

  ★ **SOLUTION:** D4 + D5 $\leq$ 1, D4 + E4 $\leq$ 1, and D5 + E4 $\leq$ 1.

  ■ **COMMON MISTAKE:** Equations containing more (or less) than 2 variables are not arc constraints. Constraints involving E5 also are not interesting as arc constraints, since the value of E5 is already determined by its node constraint. Lastly, constraints such as D4 + D5 = 1 or D4 $\neq$ D5 are not correct, since both D4 and D5 can have value 0 at the same time.

3. [4 points] If you initialized the AC-3 queue with just these three constraints, would any domain reductions result? If so, list the variables with their reduced domains. If not, give a 1-2 sentence explanation of why no reductions occur.

   ★ **SOLUTION:** No reductions would occur because if one of the variables in a constraint is set to 1, there is always the option of setting the other variable to 0 to avoid exceeding the upper limiting value of 1. If you're still confused about this, try stepping though the revise part of the AC-3 pseudocode from the text to see how it works.

4. [6 points] What if you instead initialized the AC-3 queue with the arc constraints implied by all the numbered squares (rather than just the arc constraints implied by E5)? Would AC-3 be able to figure out which 3 grid locations are the dumping locations? If so, list the dumping locations. Else, give a short explanation of why the exact locations won't be found.

   ★ **SOLUTION:** The exact locations won't be found. All AC-3 will be able to do with these constraints is reduce the domains of the squares adjacent to the 0 numbers on the grid. For example, it won't be able to reduce the domain of E2; although it can reduce the domains of D1 and D2 to 0, the constraints imposed by the 1 in E1 are just upper bounds, so they can't help AC-3 figure out that E2's value should be 1.

■ **COMMON MISTAKE:** Half credit was given for correctly identifying the 3 dumping locations: A5, D5, and E2.

5. [2 points] In AC-3 from part (4), what is the maximum number of times any variable will have its domain reduced?

   ★ **SOLUTION:** 1. Two reductions would result in an empty domain for a variable, and AC-3 would have to return false.

6. [4 points] Suppose instead of running AC-3, you decide to use DFS on this problem. If you consider only the variables A4 (with domain $\{0\}$) and A5 (with domain $\{0, 1\}$), which of these will DFS address first when using:

   - the most constrained variable heuristic?

     ★ **SOLUTION:** A4; it has the least number of variables in its domain.

   - the most constraining variable heuristic?

     ★ **SOLUTION:** A5; it participates in 4 arc constraints, while A4 participates in none.

     ■ **COMMON MISTAKE:** While the *number* in square A4 imposes constraints on the variables representing the adjacent squares, the *variable* A4 does not participate in these constraints. That is, the number in square A4 imposes the constraint A3 + B3 + B4 + B5 + A5 = 1 (or a set of 10 arc constraint equations), but the *variable* A4 does not participate in these. Thus, A4 is **not** the most constraining **variable**.

7. [2 points] If you apply the least constraining value heuristic to A5, what value should DFS assign to it first? Justify your answer with a 1 sentence explanation.

   ★ **SOLUTION:** A5 = 0 should be tried first. This results in no direct reductions on the domains of the variables that participate in arc constraints with A5, giving them the most freedom in the future.