CSCI 373
Artificial Intelligence
Midterm Exam
Fall 2011

Date	Start Time	End Time
10/ /11		

Instructions.

- Please do not open this envelope until you are ready to take the exam. Once you open it, you have up to two hours to complete it. Be sure to record your start and end times.
- Once you have completed the exam, turn it in as soon as possible. Simply slip it under my office door.
- Before taking the exam, you may prepare one 8.5 × 11 inch sheet of notes (writing on one side only), which you may use as a source during this exam.
- Aside from the 8.5 × 11 inch "cheat sheet", the only other source you may use is me. If you have questions during the exam and I am not available in my office, you may call me at (413) 441-5878 or send me email.
- Please do not discuss any aspect of the exam with anyone other than me. Even giving other students your impression of the difficulty of a problem is to be avoided.
- You should attempt to solve all problems. Partial credit will be given. Use common sense in writing your solutions, but when in doubt, more detail is probably better than less.

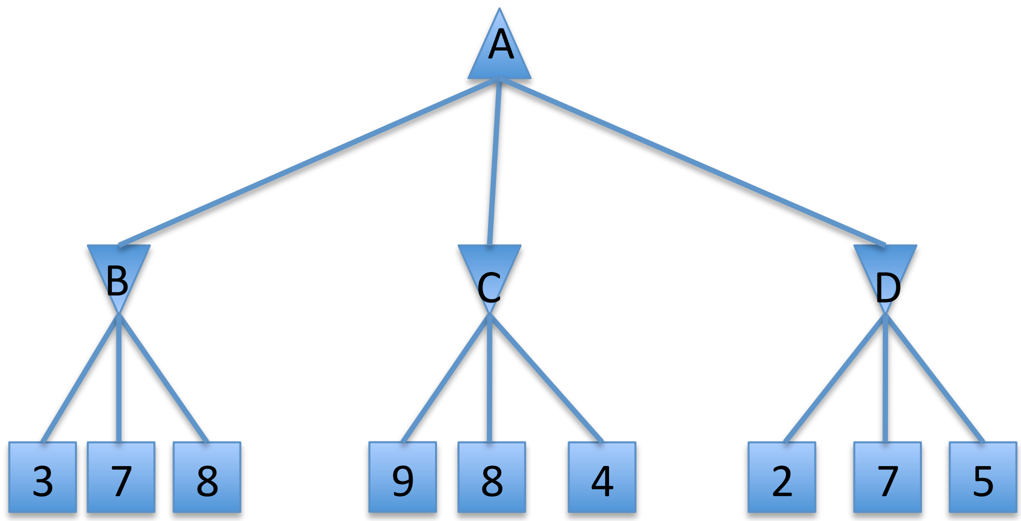
Good luck!

I have neither given nor received aid on this examination.

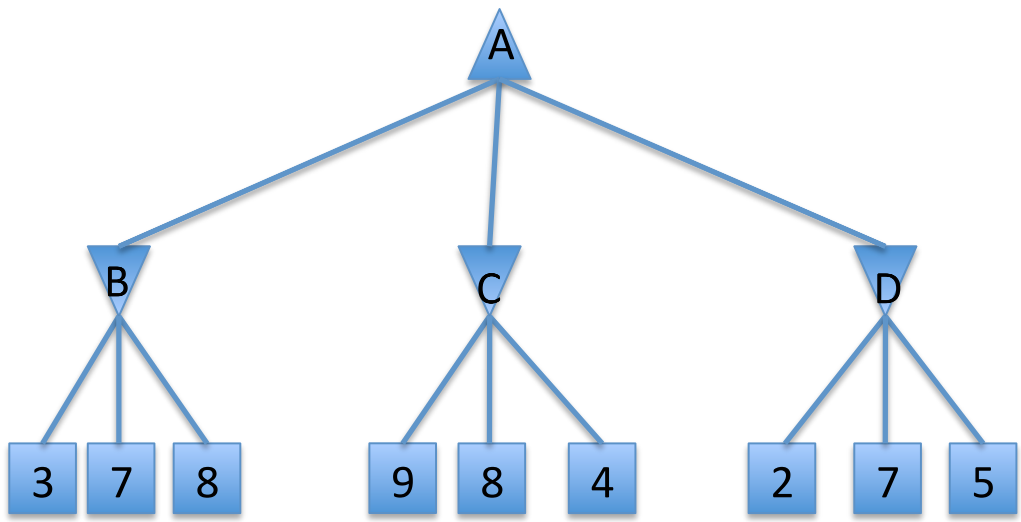
____

Question	Points	Score
1	16	
2	22	
3	22	
4	06	
5	18	
6	16	
total	100	

Problem 1 - Game Playing and Minimax (16 points)
Apply the minimax algorithm to the game tree below, where it is the *maximizer's* turn to play. The values estimated by the evaluation function are indicated at the leaf nodes. Write the estimated values of the intermediate nodes beside them, and indicate the proper move of the maximizer by circling one of the root's outgoing edges.



In the tree below, cross out the nodes that would not be evaluated if alpha-beta pruning were applied.



Reorder the nodes of the game tree given above such that alpha-beta will prune more nodes than were pruned in your previous answer. In reordering the nodes, A should remain the root, and B, C, and D should still be its children. More generally, all parent, child, and sibling relationships should be maintained.

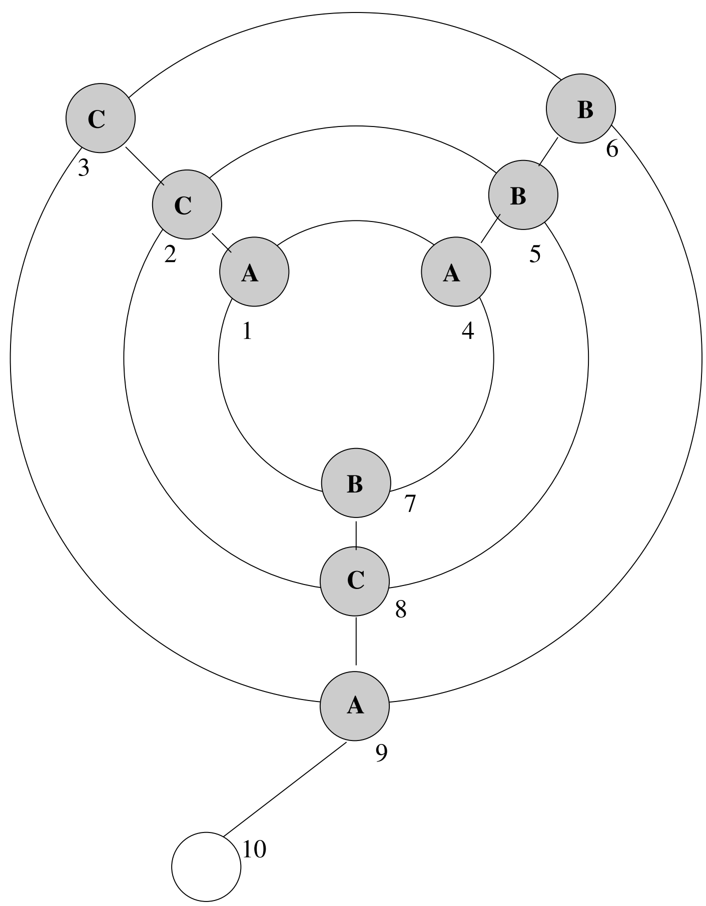
Draw the new tree below, and cross out the nodes that will be pruned.

Problem 2 - Problem Formulation as Search (22 points)

Consider the puzzle shown below. There are nine game pieces (three As, three Bs, and three Cs) that are placed in the circles labeled one through nine, as indicated.

Game pieces can be moved from one circle to another by either sliding them along the rings (say from 6 to 9 or from 7 to 4), or sliding them along the lines (for instance, from 7 to 8, from 3 to 2, or from 9 to 10), but only if the position into which they're sliding is open. At any given time, there may be at most one game piece in any of the circles labeled one through ten.

The object of the game is to arrange the pieces so as to have three different letters on each ring (1-4-7, 2-5-8, 3-6-9) and on each straight line (1-2-3, 4-5-6, 7-8-9).



Formulate this problem as a search problem. In particular, describe how you would represent and/or define each of the following. You need not write any code to answer any of the following questions. A combination of pictures and English descriptions will do. Just be sure to give enough detail to make your answers clear.

States

The Initial State

The Goal Test

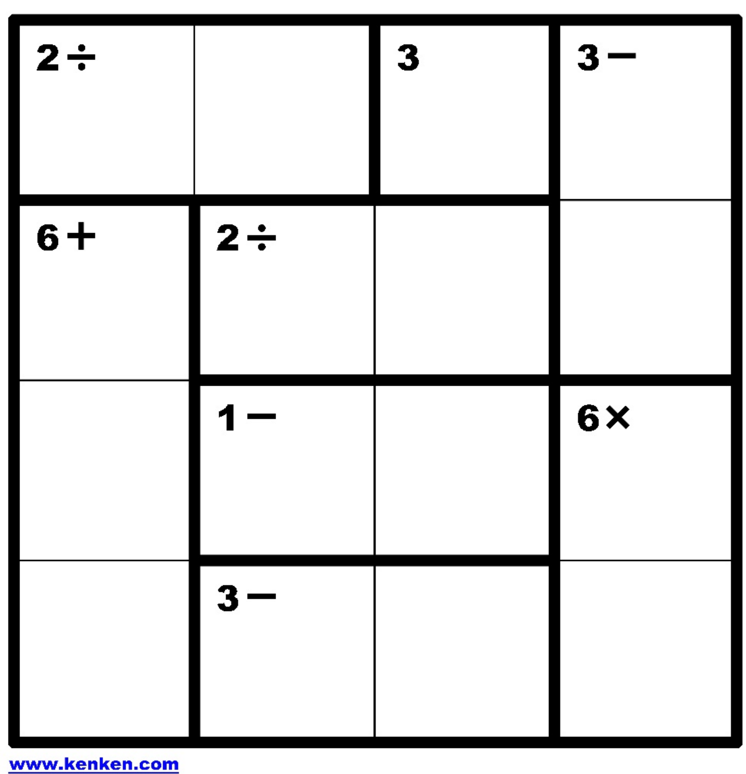
Operators

Which search algorithm would you apply to this problem? Justify your answer.

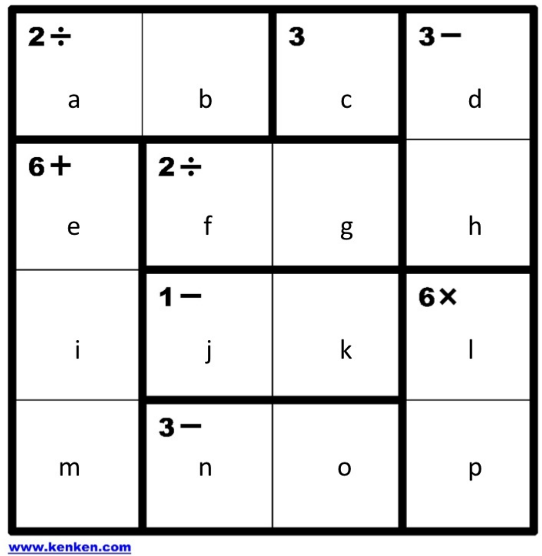
Give (and justify) an admissible heuristic for this puzzle.

Problem 3 - Constraint Satisfaction (22 points)
Below is an instance of a KenKen puzzle. If you haven't played KenKen, here are the rules for a 4×4 puzzle like the one below (www.kenken.com):

1. Fill in each grid square with a value in the range 1 to 4.
2. Do not repeat a number in any row or column.
3. The numbers in each heavily outlined set of squares, called a cage, must combine (in any order) to produce the target number in the top corner using the operation indicated.
4. Cages with one square should be filled in with the target number in the top corner.
5. A number can be repeated within a cage as long as it is not in the same row or column. (Irrelevant in our example, so ignore this for the rest of the problem.)

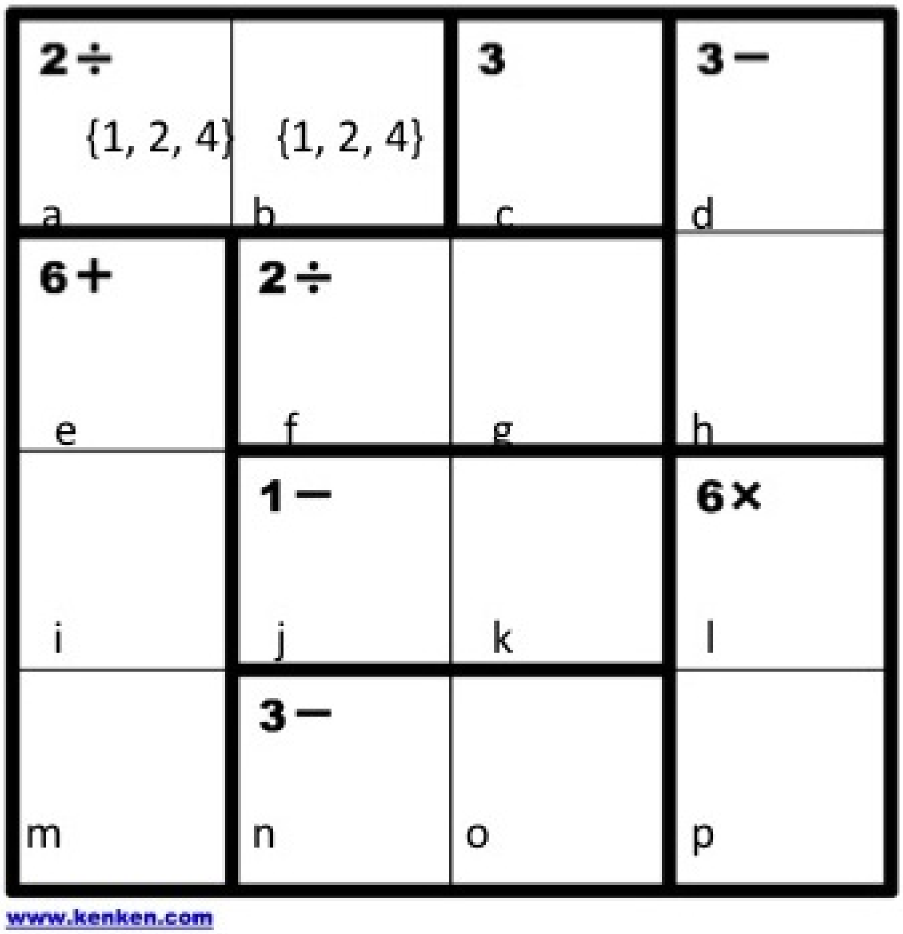


Say you'd like to approach this puzzle formally as a constraint satisfaction problem, where the goal is to assign values to the variables a through p:



List the complete set of constraints for this problem. I'll get you started: $a/b = 2$ or $b/a = 2$

Now specify the starting domain for each variable. You may begin by being clever and ruling out values that are impossible based on the arithmetic constraints you specified earlier. For instance, "$a/b = 2$ or $b/a = 2$"' limits a to the values 1, 2, and 4.



Now say you'd like to solve the puzzle by applying forward checking and backtracking.
Aside from c, which variable would you select for assignment first? Why? (Please give an answer consistent with the way one would algorithmically think about constraint propagation – not an answer based on your human intuition about this particular puzzle.)

Say that c is assigned a value first. Say that j is selected next and that the value assigned to it is 1. What is the minimum number of additional assignments that will be made before a conflict is encountered (i.e., before some variable's constraint set is emptied)? Explain.

Problem 4 - Markov Decision Processes (6 points)
Consider the 101 $\times$ 3 world shown below. In the start state the agent has a choice of two deterministic actions, *Up* or *Down*, but in the other states the agent has one deterministic action, *Right*. Assuming a discounted reward function, for what values of the discount γ should the agent choose *Up* and for which *Down*? Compute the utility of each action as a function of γ. The start state has reward 0.
Note that this simple example actually reflects many real-world situations in which one must weigh the value of an immediate action versus the potential continual long-term consequences, such as choosing to dump pollutants into a lake. [Russell and Norvig]

+50	-1	-1	-1	...	-1	-1	-1	-1
Start				...				
-50	+1	+1	+1	...	+1	+1	+1	+1

Problem 5 - Policy Iteration (18 points)
[Russell and Norvig] Consider an undiscounted MDP having three states, (1, 2, 3), with rewards -1, -2, and 0, respectively. State 3 is a terminal state. In states 1 and 2 there are two possible actions: a and b. The transition model is as follows:

- In state 1, action a moves the agent to state 2 with probability 0.8 and makes the agent stay put with probability 0.2.
- In state 2, action a moves the agent to state 1 with probability 0.8 and makes the agent stay put with probability 0.2.
- In either state 1 or state 2, action b moves the agent to state 3 with probability 0.1 and makes the agent stay put with probability 0.9.

What can be determined *qualitatively* about the optimal policy in states 1 and 2?

Apply policy iteration to determine the optimal policy and the values of states 1 and 2. Assume that the initial policy has action b in both states. Do just one round of policy evaluation and one round of policy update. In doing policy evaluation, solve for the values analytically, rather than iteratively.

What happens to policy iteration of the initial policy has action a in both states? Does discounting help? Why or why not?

Problem 6 - Miscellaneous Short Answers (16 points)
Provide brief answers to the following questions.

True or False: Iterative Deepening expands strictly more nodes than does Breadth First Search. Explain your answer.

A student has implemented both minimax and minimax-with-alpha-beta-pruning for an AI class assignment. The professor runs both and finds that the alpha-beta version consistently produces better moves than does minimax. In each case the same heuristic evaluation function is being used, and the same amount of time is allotted for deciding on a move. Assuming the student's code isn't buggy, how is it possible for the alpha-beta version to be better than minimax?

Discuss the considerations that are necessary when designing an evaluation function in the context of expectimax.

Briefly discuss ϵ-greedy selection as an exploration policy for Q-learning. What are the pros and cons of such an approach?