CSEP 573 - Artificial Intelligence
Midterm 1
Feb 7, 2019

END: 7:33 (50 mins)

p1	p2	total
p3	p4	

Name: _____

This test is closed book; no calculators or Internet is allowed. (You may bring one 8.5 x 11" piece of paper with anything written on it if you like).

If any question is ambiguous, feel free to make an assumption in order to answer it, but A) **state your assumption clearly** as part of the answer, and B) your grade will reflect the quality of the assumption.

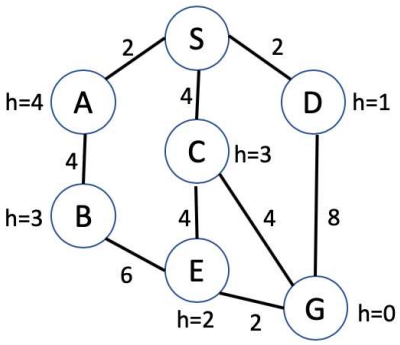
1) True / False: Circle the correct answer. (2 points each) **We'll give you one point free if you leave the answer blank** (zero points for the wrong answer), so guess with care; it can hurt your score. Some questions are tricky; read carefully!

   Remote exam: To answer the question, remove the wrong answer, i.e. leave the correct one. If you don't erase any option we consider it as no answer and give you one point.

a	**T**	Breadth-first search is complete (is guaranteed to find a solution) as long as the state space has finite branching factor.
b	**T**	Each state can only appear once in a state graph.
c	**F**	Let *b* be the branching factor of a search, *d* the depth of the solution, and *m* the maximum depth of the (finite) search space. Then the **space** complexity of **depth-first** search is b^m
d	**T**	Let *b* be the branching factor of a search, *d* the depth of the solution, and *m* the maximum depth of the (finite) search space. Then the **time** complexity of **depth-first** search is b^m
e	**T**	Let *b* be the branching factor of a search, *d* the depth of the solution, and *m* the maximum depth of the (finite) search space. Then the **space** complexity of **breadth-first** search is b^m
f	**T**	Let *b* be the branching factor of a search, *d* the depth of the solution, and *m* the maximum depth of the (finite) search space. Then the **time** complexity of **breadth-first** search is b^m
g	**T**	Let *b* be the branching factor of a search, *d* the depth of the solution, and *m* the maximum depth of the (finite) search space. Then the **time** complexity of **iterative-deepening, depth-first** search is b^m

h	**F**	A* search has worst-case space complexity that is linear in the number of states in the state graph.
i	**F**	A pattern database helps an agent avoid wasting time in cycles by storing previously-expanded states.
j	**F**	If two heuristics, h1 and h2, are admissible then their sum is admissible.
k	**T**	h(n) = 0 is an admissible heuristic for the 8-puzzle
l	**F**	Alpha-Beta pruning improves the speed of adversarial search, but on rare occasions may cause the system to miss the optimal move.
m	**F**	All the following games are zero-sum: tic-tac-toe, chess & backgammon
n	**T**	Higher values for the discount factor, γ, will, in general, cause value iteration to converge more slowly.
o	**F**	A lower value for the discount factor, γ, will cause the agent to focus on temporally distant (long term) rewards.
p	**T**	A machine learning system is said to overfit when its performance on the training set is higher than that on the validation set (or test set).

2) Search behavior. Given the graph shown to the right, write down the order in which the states are *expanded* by the following search algorithms. If a state is visited more than once, write it each time. Ties (e.g., which child to explore first in breadth-first search) should be resolved in alphabetic order (i.e. prefer A before G). Remember to include the start and goal nodes, S and G, in your answer. Assume that the algorithms execute the goal check when the nodes are visited (expanded), not when their parent is expanded to create them as children.



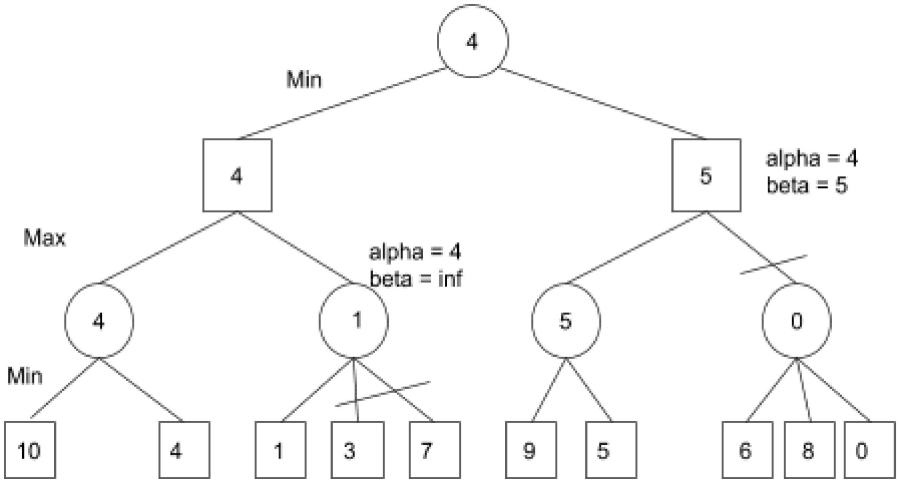
a) (3 points) Iterative deepening depth-first search (treat all edges as cost one)

S C G ____ ____ ____ ____ ____ ____ ____ ____ ____ ____ ____

b) (3 points) A* search where f(n)=g(n)+h(n) and g is the sum of the edge costs from S to n.

S C G ____ ____ ____ ____ ____ ____ ____ ____ ____ ____ ____

3) Adversarial search.
Consider the following game tree for the **minimizing player**.



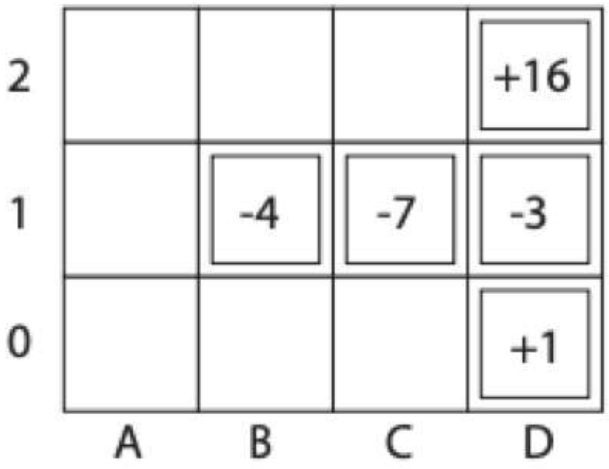
a) (2 point) Fill in the squares and circles with the backed-up values resulting from a regular minimax search.
Remote exam: Click on the picture to write inside the shapes

b) (4 points) Show how a pre-order depth-first search (from left to right) with alpha-beta pruning would work.

Draw a line thru unexplored branches.
Write down the alpha and beta values next to a node that has pruned children at the time its child nodes are pruned. (Recall that the “beta” value is the maximum score that the minimizing player is assured of and the “alpha” value is minimum score that the maximizing player is assured of.)

Remote exam: Click on the picture to draw the lines.

4) MDPs. Consider the grid world shown to the right. States with inner boxes, like D2, are terminal - no actions may be executed. In this world R(s, a, s') = $R^{dest}(s')$ and is shown as the value inside the square; so any action which causes the robot to end up in B1 accrues an immediate `reward' of -4. If no value is shown, the reward for landing in a square is zero.
For non terminal states there are four actions {N, S, E, W} comprising the compass directions. Suppose actions are deterministic, so T(A0, E, B0) = 1.0 and T(A0, E, s) = 0 for all other states s ≠ B0. Any attempt to move outside the 12 visible states results in no change. E.g., T(A2, N, A2) = 1.0
Suppose the discount factor γ = ½



a) (1 point) Suppose the robot always follows the constant policy $\pi^e(s)$ = E; in other words it always tries to move east. Calculate the value (expected discounted reward) for state A0 using this policy:

C0 = γ * D0 = 0.5, B0 = γ * C0 = 0.25, A0 = γ * B0 = 0.125

b) (3 points) Now run value iteration on the same deterministic MDP with γ = ½; compute V*(s) for all nonterminal states and fill in the grid (above) with the value of each state.
Remote exam: Please write as A0:, A1:, etc

A0 = γ * A1 = 0.5
A1 = γ * A2 = 1
A2 = γ * B2 = 2
B0 = γ * C0 = 0.25
B2 = γ * C2 = 4
C0 = γ * D0 = 0.5
C2 = γ * D2 = 8

c) (1 point) what is π^*(A0)?

π^*(A0) = N

d) (1 point) what is Q*(A0, E)?

Q*(A0, E) = 0.125

e) (2 points) Consider a modified grid world, almost exactly like the one defined by R^{dest} and T above, except where aliens occasionally intercept the robot and transport it elsewhere. Specifically, for any action, there's an 80% probability of acting as it would above, a 10% probability of ending up in state A0, and a 10% chance of ending up in a new terminal state with no reward. Write the Bellman equation for this new domain using the symbols defined for parts a-d, i.e. the original T, R^{dest}, and γ.

$V^*(s) = \max_a Q^*(s, a)$

$Q^*(s, a) = 0.8 * (R(s, a, T(s, a)) + \gamma * V^*(T(s, a)) + 0.1 * (0 + 0) + 0.1 * (0 + \gamma * V^*(A0))$

$\pi^*(s) = \text{argmax}_a Q^*(s, a)$