Practice Midterm Examination

CSCI 561 FALL2014: Artificial Intelligence

Student ID:

Last Name:

First Name:

USC email:

Rubric is not provided. You are welcome to discuss your answers to this exam on Piazza with the "midterm_exam1" tag.

You are also welcome to discuss your answers with your fellow students or groups to come up with an agreed solution. Pleas note that the actual time for Midterm1-Exam will be 90 minutes (not 80 minutes), and the actual questions in the exam may be different or harder/easier than these sample practice questions.

Instructions:

1. Date: **9/29/2014 from 5:00pm – 6:20 pm**
2. Maximum credits/points for this midterm: 100 points.
3. Credits/points for each question is indicated in the brackets [ ] before the question.
4. **No books** (or any other material) are allowed.
5. Attach extra sheets (available upon request) if required (write full name on each extra sheet).
6. **Write down name, student ID and usc email address.**
7. No questions during the exam.
8. **Be brief: a few words are often enough if they are precise and use the correct vocabulary studied in class.**
9. When finished raise completed exam sheets until approached by proctor.
10. **Adhere to the Academic Integrity code.**

1. [15%] General AI knowledge.

For each of the statements below, write **T** if the statement is always and unconditionally true, and write **F** if it is always false, sometimes false, or just does not make sense:

(i) [1%] ___ The Turing test defines the conditions under which a machine can be said to be "intelligent".

(ii) [1%] ___ My office is not an accessible environment.

(iii) [1%] ___ A contingency problem involves a nondeterministic and accessible environment.

(iv) [1%] ___ During search, one usually applies the goal test onto newly expanded children, before queuing-up these children.

(v) [1%] ___ If the cost of applying an operator once is always 1, then BFS is optimal.

(vi) [1%] ___ A* is an admissible algorithm.

(vii) [1%] ___ DFS is faster than BFS.

(viii) [1%] ___ DFS has lower asymptotic space complexity than BFS.

(ix) [1%] ___ When using the correct temperature decrease schedule, simulated annealing is guaranteed to find the global optimum in finite time.

(x) [1%] ___ Alpha-beta pruning accelerates game playing at the cost of being an approximation to full minimax.

(xi) [1%] ___ Genetic algorithms use a step called "failover".

(xii) [1%] ___ Hill-climbing is an entirely deterministic algorithm.

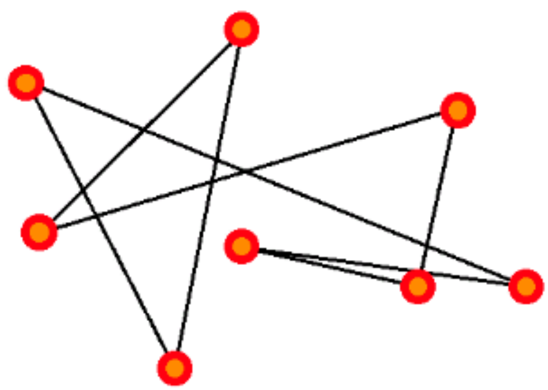
(xiii) [1%] ___ The exact evaluation function values do not affect minimax decision as long as the ordering of these values is maintained.

(xiv) [1%] ___ A perfectly rational backgammon-playing agent never loses

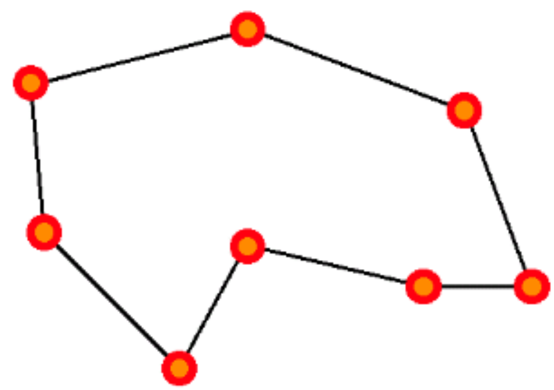
(xv) [1%] ___ Hill climbing search is best used for problem domains with densely packed goals

2. [15%] Search Algorithms Concepts.

Let's formalize the traveling salesman problem (TSP) as a search problem. Remember that the goal in this problem is to find the shortest possible route that visits every city on a map exactly once, as exemplified below:



Suboptimal solution (long path)



Optimal solution

Assume that we have *n* cities forming a set C=$\{c_1, \ldots, c_n\}$. Also assume that you can travel from any city in that set to any other city, and that the distance between any two cities c_i and c_j is given by $d(c_i, c_j)$. Please be concise but precise in your answers. Define:

(a) [5%] A suitable representation for states:

(b) [2%] The initial state of the problem:

(c) [3%] A good goal test to use in this problem:

(d) **[3%]** Good operators to use for search:

(e) **[2%]** Which search algorithm would be the most appropriate to use here if we want to ensure that the shortest possible path is found?

3. [30%] Comparing Search Strategies

Consider the search space on the following page, where S is the start node and G1, G2 and G3 satisfy the goal test. Arcs are labeled with the cost of traversing them and the estimated cost to a goal is reported inside nodes.

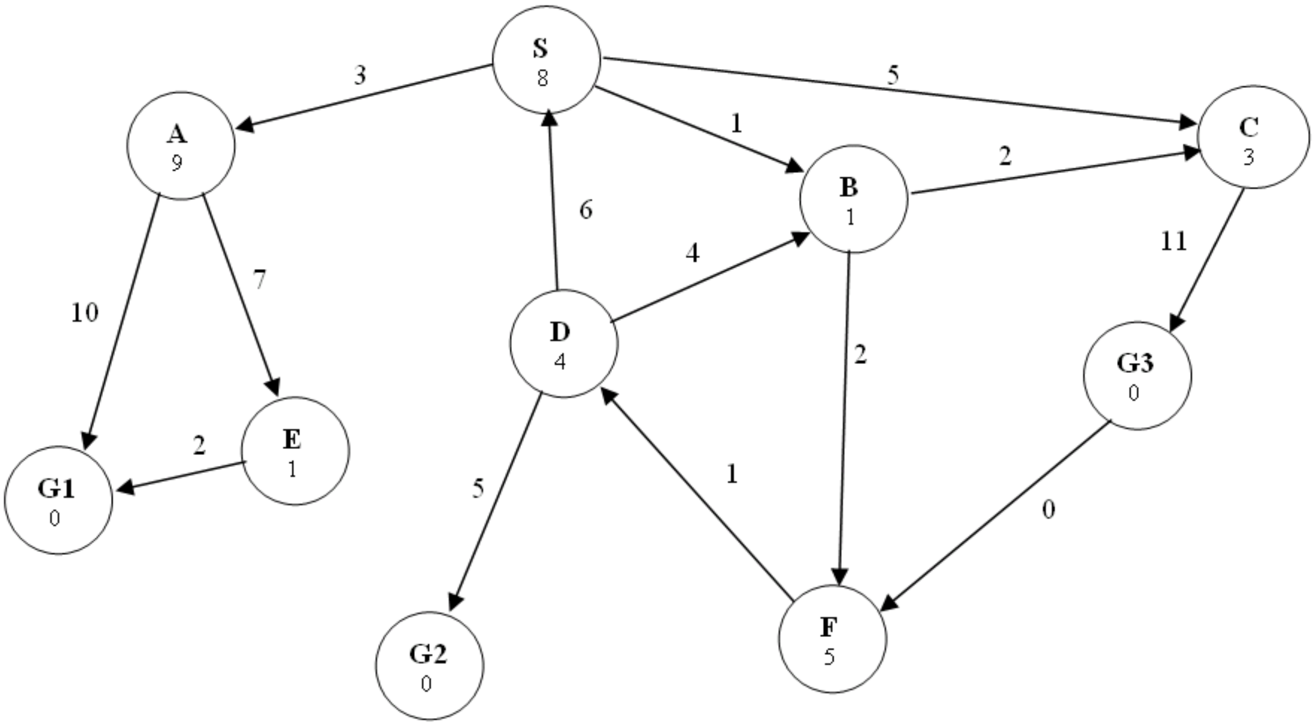
For each of the following search strategies (next page), indicate which goal state is reached (if any) and list, in order, all the states of the nodes **popped off** of the OPEN queue. When all else is equal, nodes should be removed from OPEN in alphabetical order.

You should not expand nodes with states that have already been visited. Note how the arcs in the figure are oriented, which means that you can only go from one state to another if the arrow points from the first to the second. For example, you can go from S to A (i.e., A is a successor of S) but not from A to S (i.e., S is not a successor of A).

(a) [10%] Depth-first search
Goal state reached: _______ States popped off OPEN: ________________________________

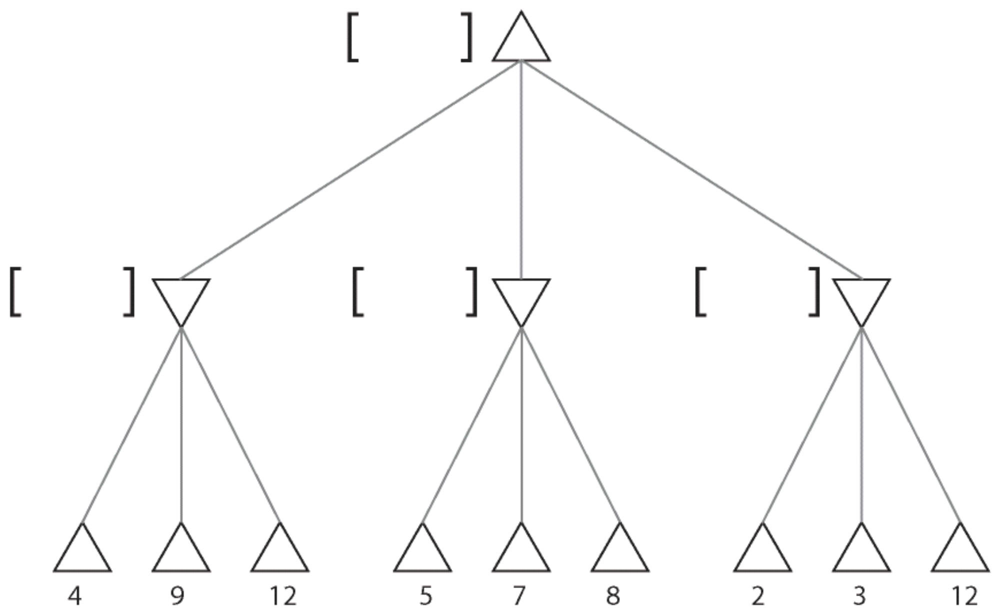
(b) [10%] Uniform cost Search
Goal state reached: _______ States popped off OPEN: ________________________________

(c) [10%] A* Search
Goal state reached: _______ States popped off OPEN: ________________________________


S
8
A
9
C
3
B
1
D
4
E
1
G1
0
G2
0
G3
0
F
5
3
5
1
2
6
4
11
10
7
2
2
5
1
0

4. [10%] Game Playing.

Consider the following game tree in which the evaluation function values are shown below each leaf node. Assume that the root node corresponds to the minimizing player. Assume that the search always visits children left-to-right.



(a) [4%] Compute the backed-up values computed by the minimax algorithm. Show your answer by writing values at the appropriate nodes in the above tree.

(b) [6%] Which nodes will not be examined by the alpha-beta pruning algorithm? Mark them on the tree above.

5. [10%] Constraint Satisfaction

Given the constraint graph for the 4-queens problem for the board below.

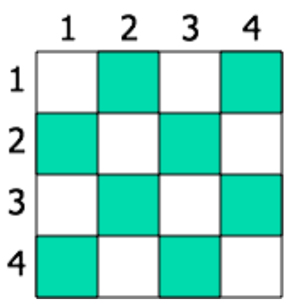
(a) [1%] List the variable names

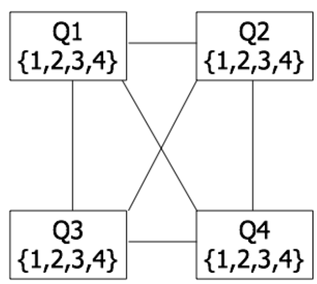
(b) [1%] What do the variables represent?

(c) [1%] What are the domain values?

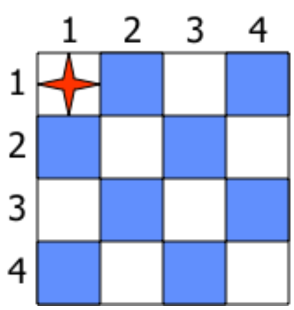
(d) [1%] How do the constraints (edges) affect the variables for this problem?

(e) [1%] How many binary constraints are there?





(f) Draw the constraint graph again after performing Arc Consistency to solve this 4 queens board, given the first queen's position is square 1.



6. [20%] AI Applications.

(a) [5%] Which AI application has a discrete, static environment?

a. **Robocup Soccer robots**
b. **Google Self-driving car**
c. **IBM's Deep Blue**
d. **All of the above**
e. **None of the above**

(b) [5%] Virtual agents Ada and Grace have this ability in common with IBM Watson?

a. **Text-to-Speech Synthesis**
b. **Natural language processing**
c. **Knowledge representation**
d. **All of the above**
e. **None of the above**

(c) [5%] In building a poker-playing robot, what is most important?

a. **It should pass the Turing Test**
b. **It should be able to hold the cards**
c. **It should be able to reason in a dynamic, continuous environment**
d. **All of the above**
e. **None of the above**

(d) [5%] An admissible heuristic for the Google Maps could be?

a. **Straight-line distance**
b. **Cost of transportation**
c. **Time during rush hour traffic**
d. **All of the above**
e. **None of the above**

[10%] MDP

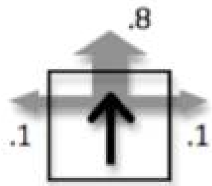
The grid world MDP shown below operates like to the one we saw in class.

The states are grid squares, identified by their row and column number (row first).

The agent always starts in state (1,1), marked with the letter S. There are two terminal goal states, (2,3) with reward +5 and (1,3) with reward -5. Rewards are 0 in non-terminal states. (The reward for a state is received as the agent moves into the state.)

The transition function is such that the intended agent movement (North, South, West, or East) happens with probability .8. With probability .1 each, the agent ends up in one of the states perpendicular to the intended direction. If a collision with a wall happens, the agent stays in the same state.

		+5
S		-5



9.a. [4%] Draw the optimal policy for this grid. Draw it directly on the grid world above.

9.b. [6%] Suppose the agent knows the transition probabilities. Give the first two rounds of value iteration updates for each state, with a discount of 0.9. (Assume V_0 is 0 everywhere and compute V_i for times i = 1, 2).

State	(1,1)	(1,2)	(1,3)	(2,1)	(2,2)	(2,3)
V_0	0	0	0	0	0	0
V_1						
V_2						