Name: _Connor Johnson_____

CS1571 Introduction to Artificial Intelligence
Midterm 2020

You have 24 hours to complete this test. It is due Tuesday, Oct. 13 at 1:15. The test has 8 pages. There are 16 questions, worth 90 points total.

The test should be submitted on **Gradescope**. Upon submission, please use the Gradescope interface to indicate where your responses to each question are on the page.

This test should represent your **individual work**. You may not discuss any parts of this test with your classmates or other people. You may not post test questions to Q&A sites like StackOverflow to receive answers during the testing period. Consultation on test questions constitutes an Academic Integrity Violation.

This test is **open book** and **open notes**. You may use any course resources you like to answer questions on this test. You may also use other resources **already available** on the internet, as long as you cite them below. Do not list all resources viewed while searching; just focus on the ones that directly contributed to the answer. Failure to cite resources you have used to compose your answer consists of an Academic Integrity Violation.

Certifications

I certify that this test represents my own individual work, and I have not consulted with others in taking the test.

___Connor Johnson_____
Signature

The following are any resources other than the course textbook or course materials I used while taking the test in order to compose my answers.

URL or Resource Name	Question Number
https://en.wikipedia.org/wiki/Markov_decision_process	15
<<add more rows as necessary>>	

I certify that the above list represents the external resources that influenced my answers on the test.

__Connor Johnson_____
Signature

A. Basic Search and Informed Search (20 pts)

1. In this course, we have talked both about the use of an explored set in search algorithms, and checking for cycles along paths in search algorithms. Which of the following is true about the use of these techniques in depth-first search (DFS). Check all that apply. (4 pts)

   ___ DFS with an explored set has better space complexity than breadth-first search with an explored set.

   _X_ DFS with checking for cycles has better in time complexity than DFS without checking for cycles.

   ___ Assume you are not maintaining an explored set, and the state space is finite. Whether you check for cycles or not determines whether DFS is complete.

   _X_ Assume the state space is finite. Whether you use an explored set or not determines whether DFS is optimal.

2. Below is a trace of a search being done on a graph. The search starts with *S* and ends with *G*, step costs are labeled directly on the graph, and the heuristic table below represents a heuristic function for the nodes. The second table ("Search Trace") represents a trace of the search being done. What kind of search is being run on the graph? One answer is correct. (4 pts)

   a. Uniform-cost search, tree search

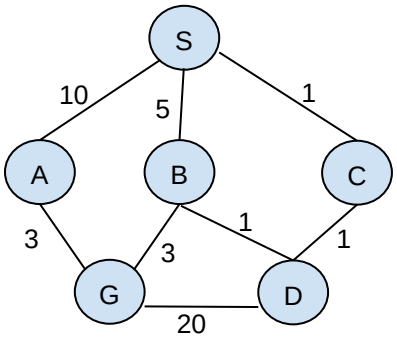
   b. Uniform-cost search, graph search

   c. Greedy search, tree search

   d. Greedy search, graph search

   e. A* search, tree search

   f. A* search, graph search – A* graph search is used below



Heuristic Table						
Parameter	h(S)	h(A)	h(B)	h(C)	h(D)	h(G)
Value	15	3	9	13	20	0

Search Trace	
Expanded	**Frontier**
S	A(3), B(9), C(13)
A	G(0), B(9), C(13)
G	

3. If you were to run an A* search on the graph in Question 3, using the same heuristic, what are its properties? Check all that apply. (4 pts)

   _X_ Complete for tree search.

   _X_ Complete for graph search.

   __ Optimal for tree search.

   __ Optimal for graph search.

4. Bidirectional search is a search algorithm not covered in the class where you run two searches: one from the start state forward, and one from the goal state backward. When the two searches meet up (i.e., a visited state in one search is equal to a visited state in the second search), you have a solution.

   Consider a breadth-first search implementation of bidirectional search (e.g., https://efficientcodeblog.wordpress.com/2017/12/13/bidirectional-search-two-end-bfs/). It is complete and optimal with uniform step costs. It has a time complexity and

space complexity of $O(b^{d/2})$, because you only have to expand half the levels of the tree in either direction before you find where the two searches will meet up. In the pseudocode from the link above (reproduced below), you check to see if the frontiers of the searches intersect, and once they do, your search is successful.

If you used two iterative deepening depth-first searches as part of the bidirectional search, instead of two breadth-first searches, how would you determine whether the two searches meet up? How does your solution influence the space complexity of the algorithm? (8 pts)

I would add an explored set for both searches that are used to check if the other search has reached a node. When one of the DFS reaches a node, it would check if that node is in the explored set of the other search. This would increase the space complexity because there are now extra sets to keep track of the nodes.

```
BIDIRECTIONAL_SEARCH
 1   Q_I.Insert(x_I) and mark x_I as visited
 2   Q_G.Insert(x_G) and mark x_G as visited
 3   while Q_I not empty and Q_G not empty do
 4       if Q_I not empty
 5           x ← Q_I.GetFirst()
 6           if x = x_G or x ∈ Q_G
 7               return SUCCESS
 8           forall u ∈ U(x)
 9               x' ← f(x, u)
10               if x' not visited
11                   Mark x' as visited
12                   Q_I.Insert(x')
13               else
14                   Resolve duplicate x'
15       if Q_G not empty
16           x' ← Q_G.GetFirst()
17           if x' = x_I or x' ∈ Q_I
18               return SUCCESS
19           forall u^-1 ∈ U^-1(x')
20               x ← f^-1(x', u^-1)
21               if x not visited
22                   Mark x as visited
23                   Q_G.Insert(x)
24               else
25                   Resolve duplicate x
26   return FAILURE
```

B. Constraint Satisfaction Problems (16 pts)

5. Imagine a constraint satisfaction problem where the variables represent different class sections and the values represent possible timeslots for the classes. The following are the domains for three variables:

   Class1 = {9AM M/W, 10:30AM M/W, 1PM M/W}
   Class2 = {9AM M/W, 10:30AM M/W, 1PM M/W, 9AM T/Th, 10:30AM T/Th, 1PM T/Th}
   Class3 = {9AM M/W, 9AM T/Th}

   Assume once a timeslot is assigned to one class, no other class can be assigned that timeslot. If you were following both the minimum remaining values and least constraining value heuristics, which assignment would be made? If there is more than one possible assignment, indicate this in your answer. (4 pts)

   Class 3 would be selected first because it has the fewest remaining timeslots. 9AM T/TTh would be selected because only Class 2 also has this timeslot while all 3 have 9AM M/W.

6. Compare the backtracking algorithm used in constraint satisfaction problems and a non-recursive DFS (such as the one discussed in class) that stores all child nodes of an expanded node. Which of the following is true? Check all that apply (4 pts)

   _X_ Backtracking has a better space complexity than the non-recursive DFS.

   _X_ Backtracking will terminate its search when it realizes it has reached a state that could not lead to a goal state, while the non-recursive DFS will not.

   ___ Backtracking always has a better time complexity than the non-recursive DFS.

   _X_ When applied to constraint satisfaction problems with a fixed number of variable assignments, backtracking is complete, while a non-recursive DFS is not.

7. Consider the following pseudocode from the textbook. The figure caption says “After applying AC-3, either every arc is arc-consistent, or some variable has an empty domain, indicating that the CSP cannot be solved.” Explain in your own words based on the pseudocode why this statement is true. (8 pts)

```
function AC-3( csp) returns false if an inconsistency is found and true otherwise
  queue ← a queue of arcs, initially all the arcs in csp

  while queue is not empty do
    (X_i, X_j) ← POP(queue)
    if REVISE(csp, X_i, X_j) then
      if size of D_i = 0 then return false
      for each X_k in X_i.NEIGHBORS - {X_j} do
        add (X_k, X_i) to queue
  return true

function REVISE( csp, X_i, X_j) returns true iff we revise the domain of X_i
  revised ← false
  for each x in D_i do
    if no value y in D_j allows (x,y) to satisfy the constraint between X_i and X_j then
      delete x from D_i
      revised ← true
  return revised
```

Figure 6.3 The arc-consistency algorithm AC-3. After applying AC-3, either every arc is arc-consistent, or some variable has an empty domain, indicating that the CSP cannot be solved. The name "AC-3" was used by the algorithm's inventor (?) because it was the third version developed in the paper.

The algorithm works by changing the value of variables at an arc then checking if the constraint is satisfied. It then moves on to another arc and keeps modifying the values of the variables until every arc is consistent. It checks every possible combinations that keep the graph arc consistent. If it is unable to find a solution, it means there is no combination of variables that keeps the arcs consistent. Thus, the CSP is not solvable

C. Adversarial Search (16 pts)

One well-known game is Connect Four. From Wikipedia (https://en.wikipedia.org/wiki/Connect_Four), Connect Four is a:

> *…two-player connection board game, in which the players choose a color and then take turns dropping colored discs into a seven-column, six-row vertically suspended grid. The pieces fall straight down, occupying the lowest available space within the column. The objective of the game is to be the first to form a horizontal, vertical, or diagonal line of four of one's own discs.*

8. Formulate the following problem as an adversarial search problem, describing the initial state, actions, transition model, and utility values at the terminal nodes. We will be grading you on your ideas rather than the elegance of your formulation. (12 pts)

   Initial State: An empty 7x6 two-dimensional array representing the board
   Actions: Place a piece in one of the 7 columns. Players must alternate actions
   Transition Model: A piece is placed in the lowest possible position in a column when a player selects that column as their action
   Utility Nodes: The terminal nodes are where the board is either full or there is 4 pieces in a row. If the board doesn't have 4 pieces in a row and is full, the utility is zero. If there is 4 pieces in a row, the utility is infinity or negative infinity if the player won or lost.

9. Assume that you want to keep move times to less than 1 seconds, and this maps to fewer than 1,000,000 expanded nodes per move. You decide to limit the depth of your search. Based on your formulation, at what level would you cut off your search tree, and how many nodes would be at that deepest level? You can give your response assuming (incorrectly) that each level has the same branching factor, equivalent to the maximum branching factor of your tree, and that you are not applying an alpha-beta pruning algorithm. (4 pts)

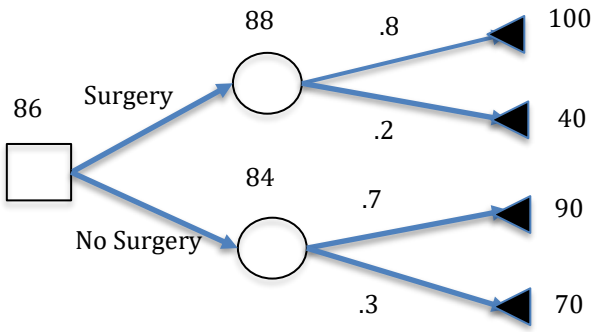
   $7^x < 1{,}000{,}000;\ x < \log_7(1{,}000{,}000);\ x < 7.10$

   I would cut off the search at a depth of 7. There would be 823,543 nodes at this depth.

D. Probability and Decision Making (20 pts)

10. The following is an example decision tree for the results of doing surgery versus not doing surgery. The values at the utility nodes represent the degree of recovery, with 100 being the best outcome. The values at the chance and decision nodes represent the potential value of taking an action at that node. There is one error in the computation made. What is it? (4 pts)

    The surgery node is a decision node, not a chance node. So the value should be the max of 88 and 84, not the average.



11. Assume that you knew that:
    P(Recovery = T) = .6
    P(Surgery = T | Recovery = T) = .8
    P(Surgery = F | Recovery = F) = .3

    What is P(Recovery = T | Surgery = T)? (4 pts)

    P(Recovery = T | Surgery = T) = .6*.8/(.6*.8 + .4*.7) = .632

12. Design and write a decision tree problem that incorporates a real-world scenario not discussed in class, and has a minimum of two sequential decision points. The problem should have two out of the three following attributes (12 pts):
    - Probabilities further down in the tree are conditioned on probabilities earlier in the tree
    - Actions yield more information about the world, but may have rewards or costs
    - A utility function assigns value to outcome nodes based on human preferences

    Write your problem in natural language, and draw the structure of the tree, and label given information (but no need to compute the values of the different nodes and edges). You will be graded on how well your problem and tree meets the requirements defined above, and if your problem and tree contain all the information needed to make a rational decision.

    You own a business with 4 employees. An employee has a friend that tested positive for COVID and that employee has a 25 percent chance of having it. A test costs you $500. Testing will reveal with accuracy whether the employee has COVID. If the employee doesn't have COVID, no loss if you don't test and $500 loss if you do. If the employee does have covid and you test, you send the employee home costing you $500 and there's a 20 percent he gave it to the other employees which would cost you another $1500. If the employee has COVID and you don't test, there is an 80 percent chance all your employees get it costing you $2000.
    Graph is on the last page

E. Reinforcement Learning and MDPs (12 pts)

13. Which of the following are differences between Q-learning and Value Iteration. Check all that apply. (4 pts)

___ Q-learning attempts to learn an optimal policy based on the utility values of world states, while Value Iteration does not.

___ Q-learning is a reinforcement learning algorithm, while Value Iteration is not.

___ Q-learning requires complete knowledge of the transition matrix, while Value Iteration does not.

_X_ In Q-learning, an agent does an action and observes what reward it gets, while in Value Iteration, the rewards of actions are known.

14. In the Bellman Equation, what does each of the following represent? Write one sentence per row. (4 pts)

$V_{k+1}(s) = \max_a \sum_{s'} Pr(s'|s,a) [R(s,a,s')+\gamma V_k(s')]$

$V_{k+1}(s)$	Value of being in a certain state
$R(s'\|s,a)$	Reward function of moving from state s to s' with action a
$Pr(s'\|s,a)$	Probability of arriving at state s' from s with action a
$\gamma V_k(s')$	Value of the state being moved into with a discount factor

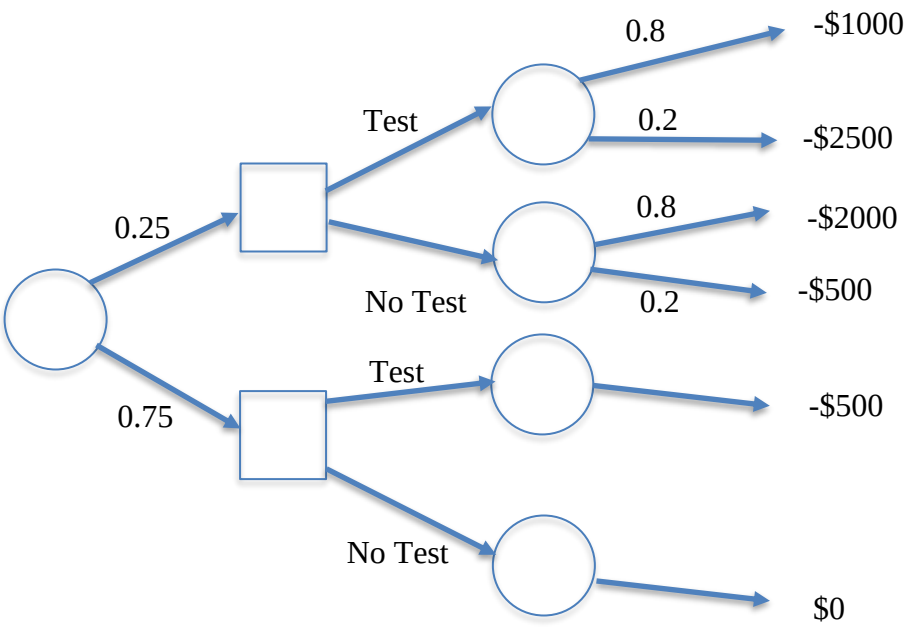
15. What is a circumstance where you would use a Markov Decision Process to represent a problem, instead of a decision tree? Your answer should relate to the features of the problem you need to model in order to decide on the best action. (4 pts)

Markov decision process is used when an action isn't guaranteed to move into a specific state. Instead, it is random with probabilities that determine the likelihood of moving into each specific state

F. Reflection (6 pts)

16. Which question on the exam do you feel like you learned the most from trying to answer, and why? (6 pts)

I learned the most from questions 13 and 15. I didn't feel like I had a good grasp on Q-Learning and Markov Decision Process when we first went over it. I reread the article and the study guide after seeing these questions which really helped me understand the topics better.


0.8
-$1000
Test
0.2
-$2500
0.25
0.8
-$2000
No Test
0.2
-$500
Test
0.75
-$500
No Test
$0