COMP3211 Fundamentals of AI **Name:** _____
Spring 2022-23 Midterm
25/03/2023 **Stu ID:** _____
Time Limit: 120 Minutes

Instructions:

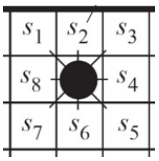
1. This exam contains 13 pages (including this cover page) and 10 questions.
2. This is a closed book exam.
3. Please write only in the exam paper.
4. You can use either pen or pencil.
5. Please have your student ID card ready for identity verification.

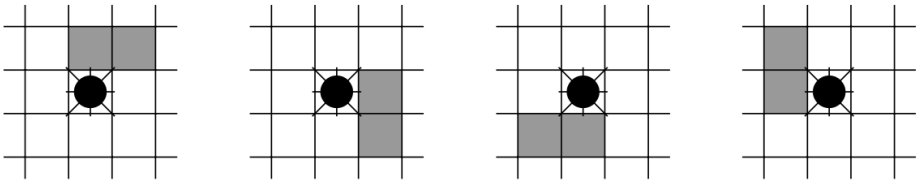
Grade Table (for teacher use only)

Question	Points	Score
1	10	
2	10	
3	10	
4	10	
5	10	
6	10	
7	10	
8	10	
9	10	
10	10	
Total:	100	

Question 1 [10 points]

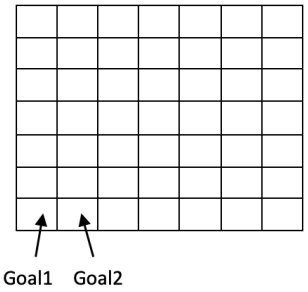
(Reactive Agents) Consider the robot with the same specification as our boundary-following robot: eight sensors, four features (see below) and four actions ($North$, $South$, $East$, and $West$).





In each diagram, the indicated feature has value 1 if and only if at least one of the shaded cells is *not* free.

1.1 Suppose the environment is given as below:



For each of the two goals indicated in the figure, can it be achieved by a reactive agent? By achieving a goal we mean that whatever the robot's initial position, the robot will get to the goal position, and then stop (*nil* action). For each of the two goals:

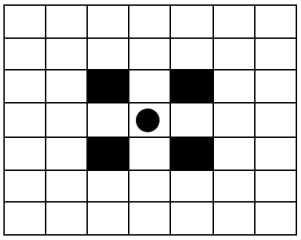
- If your answer is yes, give a production system for it. Do not call another production system. Write your own rules.
- If your answer is no, give your reason for it, and state how many steps the robot needs to remember in order to achieve the goal. No need to give a formal proof. An informal short explanation will be suffice. By k steps memory, we mean the robot remembers her past k actions and sensor readings. So 1 step memory means remembering last action and what the last sensor readings were.

1.2 The following is the production system given in the lecture note for our boundary-

following robot:

$$x_4\overline{x_1} \to north,$$
$$x_3\overline{x_4} \to west,$$
$$x_2\overline{x_3} \to south,$$
$$x_1\overline{x_2} \to east,$$
$$1 \to north.$$

Compute the next three actions according to this production system when the robot is in the middle of four obstacles as shown in the following figure:



Solution:
1.1 Goal1: Yes.
Rules: $x_3x_4 \to nil$, $x_4\bar{x}_1 \to north$, $x_3\bar{x}_4 \to west$,
$x_2\bar{x}_3 \to south$, $x_1\bar{x}_2 \to east$, $1 \to north$
Goal2: No, the agent cannot distinguish goal2 from the one on its right. The agent needs one step memory: a memory to remember goal1, and then go *East* afterward.

1.2 north north west

Question 2 [10 points]

(Perceptron) Suppose x_1, x_2 and x_3 are binary variables.

2.1 Find a perceptron (TLU) that computes the following Boolean function:

$$x_1 + x_2 + \overline{x_3}.$$

2.2 Show that there is no perceptron that can compute the following Boolean function:

$$x_1\overline{x_2} + x_2\overline{x_1}.$$

Solution: 2.1 (need to note down the inequalities or other elaborations)

$$\begin{cases} w_1 \geq \theta \\ w_2 \geq \theta \\ w_3 < \theta \\ 0 \geq \theta \end{cases} \Rightarrow \begin{cases} w_1 = 1 \\ w_2 = 1 \\ w_3 = -1 \\ \theta = 0 \end{cases} \quad \text{(could be other possibilities)}$$

2.2

$$\begin{cases} w_1 + w_2 < \theta \\ w_1 \geq \theta \\ w_2 \geq \theta \\ 0 < \theta \end{cases} \Rightarrow \text{No solution, XOR is not linear separable.}$$

Question 3 [10 points]

(Error-Correction Procedure) Given the following training set:

ID	x_1	x_2	Label
1	1	1	1
2	1	0	0
3	0	1	0
4	0	0	1

Run the error-correction procedure for 2 iterations (i.e. going through all examples for two times). Use the learning rate 0.5, and the initial weight vector $(0, 0, 0)$. Notice that the last weight is the negative of the threshold as we add a third input that is set to 1, and fix the threshold of the training TLU to be 0.

Will the procedure converge for this training set?

Solution: Error-correction procedure:

Iteration	$w_{old} = (w_1, w_2, w_3)$	$X = (x_1, x_2, x_3)$	d	sum	f	$W_{new} = (w_1, w_2, w_3)$
1	0,0,0	1,1,1	1	0	1	0,0,0
2	0,0,0	1,0,1	0	0	1	-0.5,0,-0.5
3	-0.5,0,-0.5	0,1,1	0	-0.5	0	-0.5,0,-0.5
4	-0.5,0,-0.5	0,0,1	1	-0.5	0	-0.5,0,0
5	-0.5,0,0	1,1,1	1	-0.5	0	0,0.5,0.5
6	0,0.5,0.5	1,0,1	0	0.5	1	-0.5,0.5,0
7	-0.5,0.5,0	0,1,1	0	0.5	1	-0.5,0,-0.5
8	-0.5,0,-0.5	0,0,1	1	-0.5	0	-0.5,0,0

No it will not converge as the dataset is not linearly separable.

Question 4 [10 points]

(Fitness function) Consider Problem 1 and the figure in Problem 1.1. Suppose we want to use genetic programming to evolve a controller for the robot to achieve Goal2 (in the sense given in Problem 1.1). Which of the following is the best candidate for the fitness function: given a program p, the fitness value of p is

(a) run p in 10 randomly selected starting position for 100 steps and return the total number of cells it visited in the 10 runs that are next to the boundary.

(b) run p in 10 randomly selected starting position for 100 steps and return the total number of times it visited the Goal2 cell in the 10 runs.

(c) run p in 10 randomly selected starting position for 100000 steps and return the number of times it visited the Goal2 cell in the 10 runs.

(d) run p in 10 randomly selected starting position for 100 steps, for each run r_i, let m_i be the average distance of the last 50 locations of the run to the Goal2 cell, and then return $1000 - (m_1 + \cdots + m_{10})$.

(e) run p in 10 randomly selected starting position for 100000 steps, for each run r_i, let m_i be the average distance of the last 50 locations of the run to the Goal2 cell, and then return $1000 - (m_1 + \cdots + m_{10})$.

Solution: A good fitness function needs to be accurate and efficient at the same time. The most accurate ones are (d) and (e) (once the robot reaches Goal2 cell, it should stop and stay there forever. (d) is clearly more efficient.

Question 5 [10 points]

(Supervised learning) Answer the following two questions about supervised learning:

5.1 Give two reasons why it is not a good idea to insist on computing a model (function) that agrees with all instances in the training set.

5.2 What is the difference between gradient descent and stochastic gradient descent.

Solution: **A5.1**: 1) Overfitting can damage the generalization ability of a model. 2) Outliers or noise in training set 3) Time-consuming and computationally inefficient. **A5.2**: GD uses the entire training set to make a single step in the optimization process, while SGD uses only a single point at each step to update parameters. The training of SGD on large datasets is much faster than GD, but its convergence is more noisy and fluctuating.

Question 6 [10 points]

(MDP) Consider the following modified dice game. At each round, you choose either:

- *quit* - you get \$10.
- *stay* - a 6 sided dice is rolled and if it comes up 1 or 2, then you get \$5, and the game ends, otherwise you get \$3 and the game continues to the next round.

Answer the following two questions:

6.1 Formulate this game as an MDP by defining: states, start state, end states, transition relation, and reward function. Notice that the set of actions is given: $\{quit, stay\}$.

6.2 Assuming the discount factor $\gamma = 1$, compute the optimal policy of your MDP.

Solution: A6.1 States={in, out}, Start state=in, End state=out.
Transition relation: $T(in, stay, in) = 2/3$, $T(in, stay, out) = 1/3$,
$T(in, quit, out) = 1$.
Reward function: $R_{T(in,stay,in)} = 3$, $R_{T(in,stay,out)} = 5$, $R_{T(in,quit,out)} = 10$.
A6.2 Assume policy $stay = \pi_s$, policy $quit = \pi_q$.
$Q_\pi(s, a) = \sum_i T_i(s_t, a_t, s_{t+1})(R_{T_i(s_t,a_t,s_{t+1})}) + \gamma V_\pi(s_{t+1})$
$Q_{\pi_q} = 10$
$Q_{\pi_s} = 1/3(5 + V_{\pi_s}(\text{out})) + 2/3(3 + V_{\pi_s}(\text{in})) = 11 > Q_{\pi_q}$
Therefore, the optimal policy is *stay*.

Question 7 [10 points]

(A* search) Recall the missionaries and cannibals problem:

> In the missionaries and cannibals problem, three missionaries and three cannibals must cross a river using a boat which can carry at most two people, under the constraint that, for both banks, if there are missionaries present on the bank, they cannot be outnumbered by cannibals (if they were, the cannibals would eat the missionaries). The boat cannot cross the river by itself with no people on board.

This problem can be formulated as a search problem as follows:

- States: tuples (m, c, b), where $m, c \in \{0, 1, 2, 3\}$, $b \in \{0, 1\}$, $m \geq c$ if $m \neq 0$, and $3 - m \geq 3 - c$ if $3 - m \neq 0$. The meaning of a state is that m and c are the number of missionaries and cannibals, respectively, on the left bank, and if $b = 0$, then the boat is at the left bank, otherwise, it is at the right bank.
- Initial state: $(3, 3, 0)$.
- Goal condition: $(0, 0, b)$ for any b.
- Operators: $move(m, c)$ - move m missionaries and c cannibals from the bank where the boat is to the other side, provided $m + c \leq 2$. Notice that all states are legal. So if this action were to cause some missionaries being eaten, then it could not be applied.
- Operator cost: each operator has one unit of cost.

Let h be the following heuristic function: $h(m, c, b) = m + c$, i.e. the heuristic value of a state is the sum of the missionaries and cannibals on the left bank. Answer the following questions:

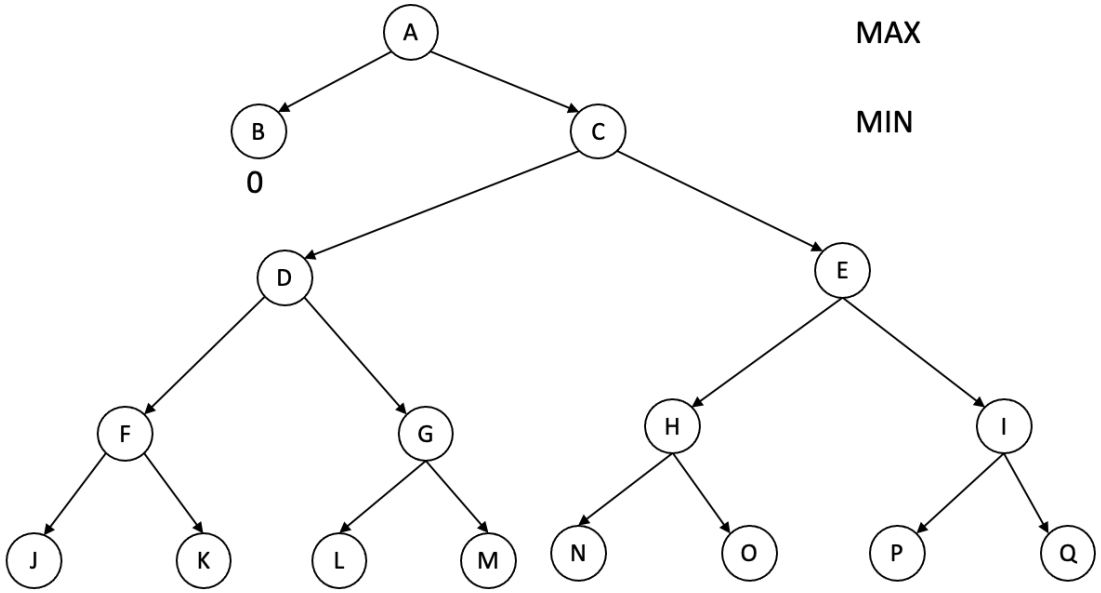
7.1 Is this heuristic function h admissible? Explain your answer.

7.2 Use this heuristic function to carry out two steps of $A*$ search-by-tree starting with the root node (which is created using the initial state): 1. expand the root, and indicate which node will be chosen for expansion next; 2. expand the chosen node and indicate which node will be chosen for expansion next. You can use any tie-breaking rule. You can answer the question by OPEN list sequence or by drawing a search tree with the chosen nodes clearly marked.

Solution: **A7.1**: h is not admissible. Because node $(1, 1, 0)$ only need 1 move to reach goal, but $h(1, 1, 0) = 2$ overestimates the cost.
A7.2: Tie-breaking rule is reducing m first.
Step1: OPEN=[(3,3,0)], CLOSE=[], f=[6]
Step2: OPEN=[(2,2,1),(3,1,1),(3,2,1)], CLOSE=[(3,3,0)], f=[5,5,6]
Step3: OPEN=[(3,2,0),(3,1,1),(3,2,1)], CLOSE=[(2,2,1),(3,3,0)], f=[7,5,6]
Step4: OPEN=[(3,2,0),(3,2,1)], CLOSE=[(3,1,1),(2,1,1),(3,3,0)], f=[7,6]
For step2, we expand node (2,1,1); For step3, we expand node (3,1,1).

Question 8 [10 points]

(Game tree search) Consider the following game tree:



For alpha-beta pruning with left to right ordering, answer the following questions:

8.1 If J's value is 1, and K's value is -1, will K be pruned? Explain your answer.

8.2 If J's value is -1, and K's value is 1, will K be pruned? Explain your answer.

Solution:

8.1 No.
$B = 0$, therefore, for node A, $\alpha = 0$.
$J = 1, K = -1$, for node F, $\alpha = 0, \beta = 1$.
$\alpha < \beta$, therefore, K cannot be pruned.

8.2 Yes.
$B = 0$, therefore, for node A, $\alpha = 0$.
$J = -1, K = 1$, for node F, $\alpha = 0, \beta = -1$.
$\alpha \geq \beta$, therefore, K can be pruned.

Question 9 [10 points]

(Search problem formulation) Consider the Tic-Tac-Toe game. Suppose you're 'X' so you move first. Suppose you know the other player's strategy which is to mark the first open position by 'O' according to the numbers indicated below (lower numbers have higher priority):

1	2	3
4	5	6
7	8	9

You now want to compute your plan for winning the game starting with the empty board. Answer the following two questions:

9.1 Formulate this problem as a search problem by defining: states, the initial state, goal condition, operators and their costs.

9.2 Is there a solution of your search problem? If so, give one solution. If not, explain why the goal is not achievable.

Solution:

9.1
- States: The state of the game is represented by a 3x3 matrix where each cell can have a value of 'X', 'O', or ' ' (empty).
- Initial state: The initial state is an empty 3x3 matrix.

- Goal Condition: The goal is to have three 'X's in a row, column, or diagonal.
- Operators: The operator is placing an 'X' in any of the empty cells.
- Costs: The cost of each operator is 1.

9.2 Yes, place X in 7, 8, 9 while O is in 1, 2.

Question 10 [10 points]

(Reinforcement learning) Consider again the Tic-Tac-Toe game. Suppose you are 'X', and you do not know the other player's strategy, and you want to play this game for tie, meaning you get positive point, say 1, if the game results in a tie, and negative point -1 otherwise. Use reinforcement learning to learn a good strategy to play this game by answering the following questions:

10.1 Define the states, available actions, starting state, end state condition, reward function.

10.2 Give two example episodes of this game of the form:

$$\begin{aligned} &\text{Episode 1:} \quad S_0, A, r_1, S_1, ... \\ &\text{Episode 2:} \quad S_0, A, r_2, S_2, ... \end{aligned}$$

where S_0 is the starting state, A the action that mark the center with 'X', and S_1 and S_2 must be different states, and r_1 and r_2 are rewards that you defined in the previous step.

10.3 Use Monte-Carlo method to estimate $Q(S_0, A)$ using the two episodes. Assume $\gamma = 1$.

Solution:

10.1
- States: all positions with equal number of X's and O's on the board.
- Actions: place an X in any of the empty cells.
- Starting state: an empty grid.
- End state condition: The game ends when either one of the players gets three in a row, or when the grid is filled without a winner.
- Reward function:
  - +1 if the game ends in a tie.
  - -1 if the game ends with a win for 'X' or 'O'.
  - 0 if the game is still in progress.

10.2
- Episode 1:
  - S_0 = empty grid,
  - A = place X in (2,2),
  - $r_1 = 0$,
  - S_1 = [O - -][- X -][- - -],
  - A = place X in (1,2),
  - $r_2 = 0$,
  - S_2 = [O X O][- X -][- - -],
  - A = place X in (3,2),
  - r_3 = -1,
  - S_3 = [O X O][- X -][- X -],
  - X wins.
- Episode 2:
  - S_0 = empty grid,
  - A = place X in (2,2),
  - $r_1 = 0$,
  - S_1 = [- - O][- X -][- - -],
  - A = place X in (1,1),
  - $r_2 = 0$,
  - S_2 = [X - O][- X O][- - -],
  - A = place X in (1,2),
  - r_3 = -1,
  - S_3 = [X X O] [- X O] [- - O],
  - O wins.

10.3 $Q(S_0, A) = (0 + 0 + (-1) + 0 + 0 + (-1))/2 = -1$

Production Systems

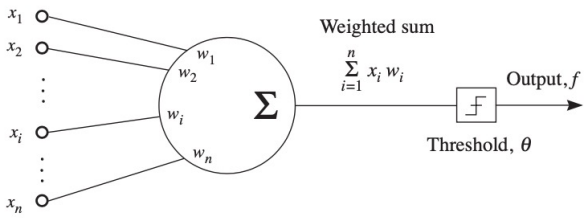
One convenient representation for an action function is *production systems*.

- A production system is a *sequence* of *productions*, which are rules of the form:

  $$c \rightarrow a$$

  meaning if condition c is true, then do a. Here a could be a primitive action, a call to a procedure, or a set of actions to be executed simultaneously.
- When there are more than one productions can be fired (their condition parts are true), then the first one is applied.

A type of circuits of particular interest in AI is *threshold logic unit (TLU)*, also called *perceptron*:



$$f = 1 \text{ if } \sum_{i=1}^{n} x_i w_i \geq \theta$$
$$= 0 \text{ otherwise}$$

The Error-Correction Procedure

Given a vector $\boldsymbol{X}$ from the training set (augmented by the $n+1$th special input 1), let d be the label (desired output) of this input, and f the actual output of the old TLU, the weight change rule is:

$$\boldsymbol{W} \leftarrow \boldsymbol{W} + c(d - f)\boldsymbol{X},$$

where c, a number, is called the learning rate.

A* Search by Tree

A^* search by trees - check only ancesters for repeated states

1. create a search tree T, consisting solely of the start node, n_0. Put n_0 on a list called *OPEN*.
2. If *OPEN* is empty, then exit with failure.
3. Select the first node on *OPEN*, and remove it from *OPEN*. Call this node n.
4. If n is a goal node, exit successfully with the solution corresponding to the path from root n_0 to this node.
5. Expand node n, generating the set M of its successors that are not already ancestors of n in G. Install these members of M as children of n in G, and add them to *OPEN*.
6. reorder the list *OPEN* in order of increasing $g(n) + h(n)$ values. (Ties are resolved in favor of the deepest node in the search tree.)
7. go to step 2.

Monte Carlo Method - Q values

$Q_\pi(s, a)$ - the expected utility in s when taking action a and then following π.
Samples:

$$D = [s_1, a_1, r_1, s_2, a_2, r_2, s_3, ...]$$

Utility at s_i:

$$u_i = r_i + \gamma r_{i+1} + \gamma^2 r_{i+2} + ...$$

Estimate:

$$\hat{Q}_\pi(s, a) = \text{average of } \{u_t \mid s_t = s, a_t = a\}$$

Extra Blank Page