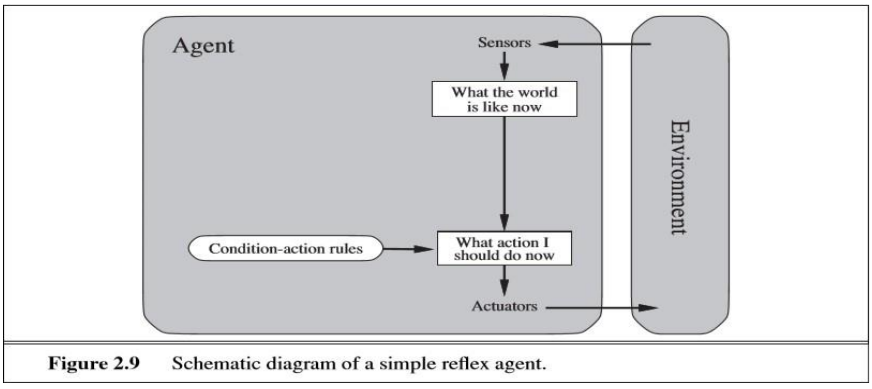
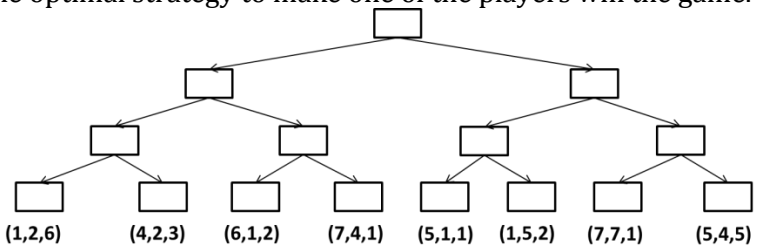
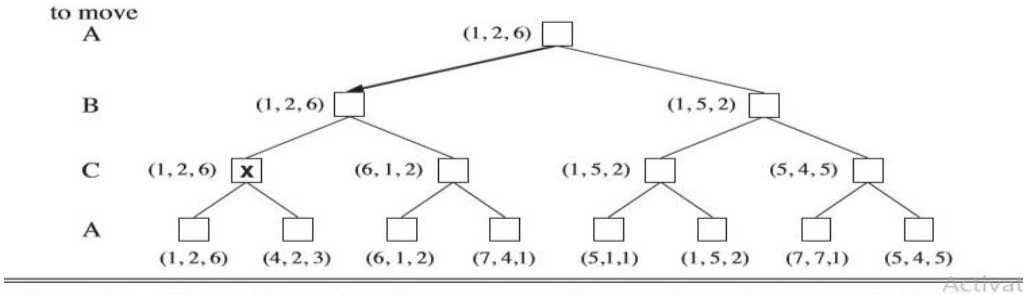
NITTE MEENAKSHI INSTITUTE OF TECHNOLOGY
(AN AUTONOMOUS INSTITUTE AFFILIATED TO VTU, BELAGAVI)

Fifth Semester Mid-Semester Examination BE Degree (MSE-I)
Academic year 2023-2024
Department of Computer Science and Engineering
Artificial Intelligence and Machine Learning (21CSG53)
MSE-I SCHEME & SOLUTION

Question No		Question	Marks
Answer the following questions.			
1.	a.	Outline the four categories of AI and mention the significance of each category.	2
	Sol:	**0.5M for each category ---------- 4 * 0.5M = 2M** The four categories of AI are as follows: 1. Thinking Humanly 2. Thinking Rationality 3. Acting Humanly 4. Acting Rationally **Significance:** • The definitions of Thinking Humanly and Thinking Rationality are concerned with thought processes and reasoning. • The definitions of Acting Humanly and Acting Rationally are address behaviour. • The definitions of Thinking Humanly and Acting Humanly measure success in terms of fidelity to human performance. • The definitions of Thinking Rationally and Acting Rationally measure against an ideal performance measure called Rationality.	**4 * 0.5M = 2M**
	b.	Illustrate the Simple Reflex agent with a neat diagram and write the pseudocode for the same.	2
	Sol:	**Diagram – 1M** **Pseudocode – 1M**  **Figure 2.9** Schematic diagram of a simple reflex agent. **function** SIMPLE-REFLEX-AGENT(*percept*) **returns** an action **persistent**: *rules*, a set of condition–action rules *state* ← INTERPRET-INPUT(*percept*) *rule* ← RULE-MATCH(*state*, *rules*) *action* ← *rule*.ACTION **return** *action* **Figure 2.10** A simple reflex agent. It acts according to a rule whose condition matches the current state, as defined by the percept.	**1M** **1M**

	c.	Consider the 3-ply Game tree with the players A, B C and utility values given in Fig 1(c). Find the optimal strategy to make one of the players win the game.  Fig 1(c): Game Tree	2
	Sol:	 **Figure 5.4** The first three plies of a game tree with three players (A, B, C). Each node is labeled with values from the viewpoint of each player. The best move is marked at the root.	**2M**
		UNIT-I	
2	a.	List the properties of a Task environment. Interpret the suitable properties for the following Task environments: i. Crossword puzzle ii. Medical Diagnosis	5
	Sol:	The properties of Task environment are as follows: • Fully observable vs. partially observable • Single agent vs. multiagent • Deterministic vs. stochastic • Episodic vs. sequential • Static vs. dynamic • Discrete vs. continuous • Known vs. unknown **1M** **1M** **2 * 2M = 4M** **Crossword puzzle – 2M** **Medical Diagnosis – 2M**	

Properties	Crossword puzzle	Medical Diagnosis
Observable	Fully	Partially
Agents	Single	Single
Deterministic	Deterministic	Stochastic
Episodic	Sequential	Sequential
Static	Static	Dynamic
Discrete	Discrete	Continuous
Known / Unknown	Unknown	Unknown

	b.	Greedy Best First Search tries to expand a node that is closest to the goal, on the grounds that this is likely to lead to a solution quickly, whereas the A* search algorithm tries to minimize the total estimated solution cost. Using the tree given in Fig 2(b), which depicts the scenario of the travelling salesperson starts from city 1 and should reach city 8. The values on edges represent the cost of reaching from one city to another. Analyze the steps to reach the goal node and find the final cost using A* Search algorithms. Use heuristic value provided in Table 2(b) suitably.	5

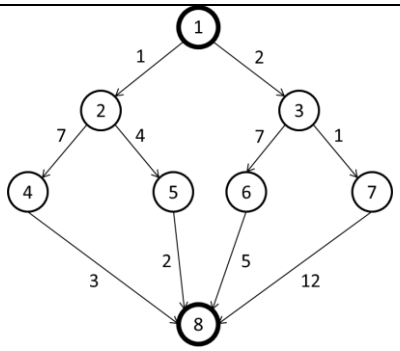


Fig 2(b): Tree

Node	H(n)
1	10
2	5
3	6
4	4
5	15
6	5
7	8
8	0

Table 2(b): Heuristic Values

Sol: **Formula for A* Search algorithm is f(n) = g(n) + h(n)**

Step 1	1 -> 2	f(n) = 1+5 = 6	✓
	1 -> 3	f(n) = 2+6=8	
Step 2	1 -> 2 -> 4	f(n) = 8+4=12	✓
	1 -> 2 -> 5	f(n) = 5+15=20	
	1 -> 3 -> 6	f(n) = 9+5=14	
	1 -> 3 -> 7	f(n) = 3+8=11	
Step 3	1 -> 2 -> 4 -> 8	f(n) = 11+0=11	
	1 -> 2 -> 5 -> 8	f(n) = 7+0=7	✓
	1 -> 3 -> 6 -> 8	f(n) = 14+0=14	
	1 -> 3 -> 7 -> 8	f(n) = 15+0=15	

Optimal path is 1 -> 2 -> 5 -> 8 with cost 7

5

OR

3 a. Describe how to formulate the 8-Queens problem with 5 components by applying the incremental formulation and complete state formulation. 5

Sol:

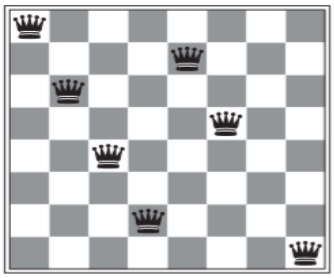
2 types of formulation – 1M each ---------- 2 * 1M = 2M
Diagram – 0.5M
Five components – 5 * 0.5M = 2.5M

- The goal of the 8-queens problem is to place eight queens on a chessboard such that no queen attacks any other. Figure shows an attempted solution that fails: the queen in the rightmost column is attacked by the queen at the top left.

There are two main kinds of formulation

- An **incremental formulation** involves operators that augment the state description, starting with an empty state; for the 8-queens problem, this means that each action adds a queen to the state.
- A **complete-state formulation** starts with all 8 queens on the board and moves them around. In either case, the path cost is of no interest because only the final state counts.

2M

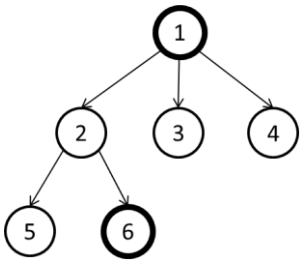
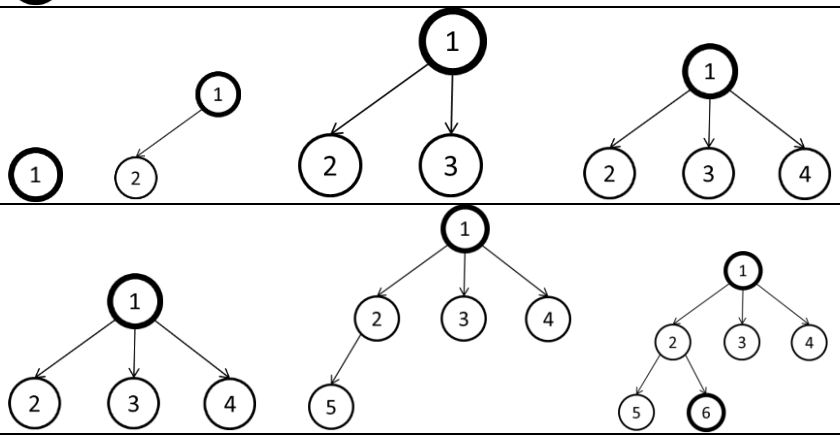


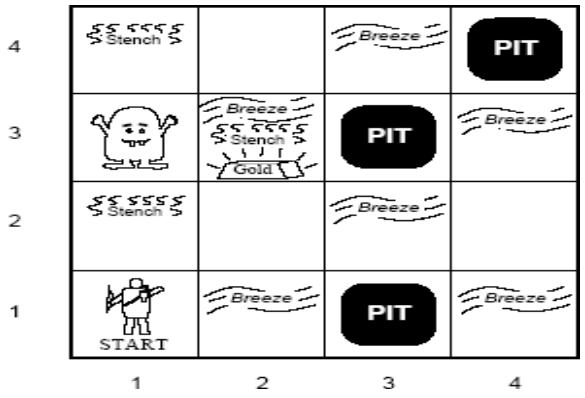
0.5M

The first incremental formulation:

- **States:** Any arrangement of 0 to 8 queens on the board is a state.
- **Initial state:** No queens on the board.

5 * 0.5M

		• **Actions:** Add a queen to any empty square. • **Transition model:** Returns the board with a queen added to the specified square. • **Goal test:** 8 queens are on the board, none attacked	**= 2.5M**
	b.	Write the Iterative deepening depth-first search algorithm. Using the same, provide the solution for the Binary Tree depicted in Fig 2(b) where Initial state is node 1 and Goal state is node 6.  Fig 2(b): Binary Tree	5
	Sol:	**Algorithm – 2M** **Problem solving – 3M (1M for each level)** **function** ITERATIVE-DEEPENING-SEARCH(*problem*) **returns** a solution, or failure **for** *depth* = 0 **to** ∞ **do** *result* ← DEPTH-LIMITED-SEARCH(*problem*, *depth*) **if** *result* ≠ cutoff **then return** *result* **Figure 3.18** The iterative deepening search algorithm, which repeatedly applies depth-limited search with increasing limits. It terminates when a solution is found or if the depth-limited search returns *failure*, meaning that no solution exists. Level 0 Level 1 Level 2 	**2M** **3M**
		UNIT-II	
4	a.	The wumpus world is a cave consisting of rooms connected by passageways. Lurking somewhere in the cave is the terrible wumpus, a beast that eats anyone who enters its room. The wumpus can be shot by an agent, but the agent has only one arrow. Some rooms contain bottomless pits that will trap anyone who wanders into these rooms (except for the wumpus, which is too big to fall in). The only mitigating feature of this bleak environment is the possibility of finding a heap of gold. Provide the PEAS description for this environment.	5
	Sol:	**PEAS description – 4* 1M = 4M** **Representative Diagram – 1M**	

		Performance measure: • +1000 for climbing out of the cave with the gold, • –1000 for falling into a pit or being eaten by the wumpus, • –1 for each action taken and –10 for using up the arrow. • The game ends either when the agent dies or when the agent climbs out of the cave. **Environment:** • A 4×4 grid of rooms. • The agent always starts in the square labeled [1,1], facing to the right. • The locations of the gold and the wumpus are chosen randomly, with a uniform distribution, from the squares other than the start square. • In addition, each square other than the start can be a pit, with probability 0.2. **Actuators:** • **go forward** • **turn right** 90 degrees • **turn left** 90 degrees • **grab**: Pick up an object that is in the same square as the agent • **shoot**: Fire an arrow in a straight line in the direction the agent is facing. The arrow continues until it either hits and kills the wumpus or hits the outer wall. The agent has only one arrow, so only the first Shoot action has any effect • **climb** is used to leave the cave. This action is only effective in the start square • **die**: This action automatically and irretrievably happens if the agent enters a square with a pit or a live wumpus **Sensors:** The agent has five sensors, each of which gives a single bit of information: – • In the square containing the wumpus and in the directly (not diagonally) adjacent squares, the agent will perceive a **Stench**. • In the squares directly adjacent to a pit, the agent will perceive a **Breeze.** • In the square where the gold is, the agent will perceive a **Glitter**. • When an agent walks into a wall, it will perceive a **Bump.** • When the wumpus is killed, it emits a woeful **Scream** that can be perceived anywhere in the cave.  	1M 1M 1M 1M 1M
	b.	Compute the optimal strategy by applying Alpha-Beta Pruning for the Game Tree depicted in Fig 3(b) with utility values.	5

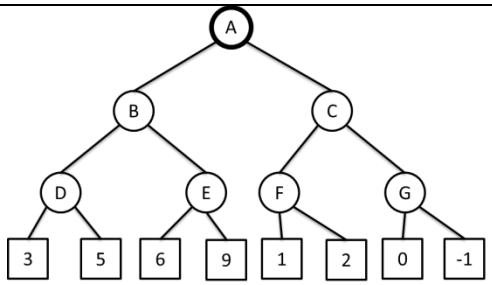
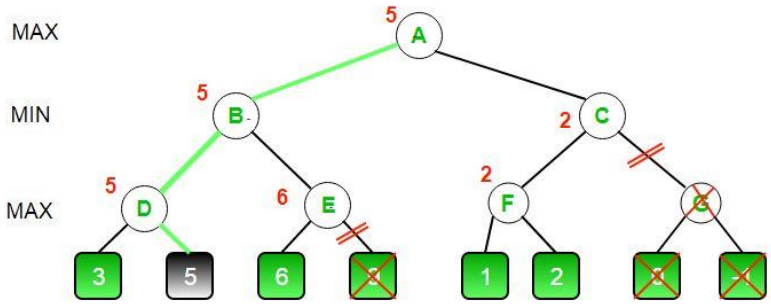


Fig 3(b): Game Tree

Sol: **Step by step calculations – 3M**
Final answer – 2M



5M

OR

5 a. Consider a 2-player game where both the players play optimally from their end to win the game. Write the Minimax algorithm to determine the optimal strategy by using the utility values at its terminal states. Also, mention the Time and Space complexity for the same. — 5

Sol: Algorithm – 4M
Time complexity – 0.5M, Space complexity – 0.5M

```
function MINIMAX-DECISION(state) returns an action
  return arg max_{a ∈ ACTIONS(s)} MIN-VALUE(RESULT(state, a))

function MAX-VALUE(state) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← −∞
  for each a in ACTIONS(state) do
    v ← MAX(v, MIN-VALUE(RESULT(s, a)))
  return v

function MIN-VALUE(state) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← ∞
  for each a in ACTIONS(state) do
    v ← MIN(v, MAX-VALUE(RESULT(s, a)))
  return v
```

Figure 5.3 An algorithm for calculating minimax decisions. It returns the action corresponding to the best possible move, that is, the move that leads to the outcome with the best utility, under the assumption that the opponent plays to minimize utility. The functions MAX-VALUE and MIN-VALUE go through the whole game tree, all the way to the leaves, to determine the backed-up value of a state. The notation $\text{argmax}_{a \in S} f(a)$ computes the element a of set S that has the maximum value of $f(a)$.

4M

- **Time complexity-** As it performs DFS for the game-tree, so the time complexity of Min-Max algorithm is **$O(b^m)$**, where b is branching factor of the game-tree, and m is the maximum depth of the tree. — 0.5M
- **Space Complexity-** Space complexity of Mini-max algorithm is also similar to DFS which is **O(bm)**. — 0.5M

	b.	Elucidate the components of a Constraint Satisfaction Problem (CSP). Solve the following Cryptarithmetic problem using CSP. C R O S S + R O A D S D A N G E R	5
	Sol:	**CSP components – 2M** **Problem solving – 3M** A constraint satisfaction problem consists of three components, V, D and C: • V is a set of variables, {V1,...,Vn}. • D is a set of domains, {D1,...,Dn}, one for each variable. • C is a set of constraints that specify allowable combinations of values. CROSS 9 6 2 3 3 ROADS 6 2 5 1 3 ---------- ---------- DANGER 1 5 8 7 4 6	**2M** **3M**
6	a.	Consider a smart vacuum cleaner agent that cleans rooms size of n * n. The Agent can move Up, Down, Left and Right. Design a program to simulate the working of the Vacuum cleaner agent and calculate the overall performance.	4

	Sol:	import random
		def display(room):
		print(room)
		room = [
		[1, 1, 1, 1],
		[1, 1, 1, 1],
		[1, 1, 1, 1],
		[1, 1, 1, 1],
		]
		print("All the rooom are dirty")
		display(room)
		
		x =0
		y= 0
		
		while x < 4:
		while y < 4:
		room[x][y] = random.choice([0,1])
		y+=1
		x+=1
		y=0
		
		print("Before cleaning the room I detect all of these random dirts")
		display(room)
		x =0
		y=0
		z=0
		
		while x < 4:
			1M
			
			1M
			
			4M

```
  while y < 4:
    if room[x][y] == 1:
      print("Vaccum in this location now,",x, y)
      room[x][y] = 0
      print("cleaned", x, y)
      z+=1
   y+=1
  x+=1
  y=0
pro= (100-((z/16)*100))
print("Room is clean now, Thanks for using")
display(room)
print('performance=',pro,'%')
```

1M

1M

Faculty Signature	*HOD Signature*