Nikita Chhabria
Roll no. 07
D15C

AIDS Assignment 1

04/05

Q1 What is AI?

→ Artifical Intelligence refers to the simulation of human intelligence in machines AI systems can perform tasks such as learning, reasoning problem solving, perception and language understanding.

AI is categorised into various types - including narrow & general AI

Narrow AI : specialised for specific tasks
example :- Siri, Google Assistant

general AI : Hypothetical AI capable of human like reasoning across domains

Q2 What are AI agents terminology, explain with example.

→ An AI agent is an entity that provides perceives its environment through sensors & acts upon its perception action cycle to achieve a goal.

AI agents can be classified as:

1. Simple Reflex Agents : Reacts to current percepts
2. Model Based Agents : Maintain internal model

Q2

(3) Goal Based Agents : Maintains internal models

(4) Utility based :- Maximizes utility functions

The example of each of the different agents is that :

Simple Reflex Agents (Rule-Based)

Thermostat :- Turns on/off heat based on temperature

Model-Based Reflex Agents

Self driving cars : Use sensor data to track obstacles & road conditions

Utility-Based Agents

Stock marketing bots : Optimize for maximum profit while managing risk

Learning Agent :

Recommendation System :- Learn from preference

Q1 AI plays a crucial role in various aspects of healthcare & daily life, such as:

- Early detection & diagnosis : AI-based models detected and analyzed patterns
- Medical Research & Drug Discovery : It discovers about potential treatment
- Healthcare automation : Supply chain automation & remote learning are few other ways AI helped during the pandemic

Q2 How AI agent used to solve 8 puzzle problem?

→ The 8-puzzle problem is a sliding puzzle consisting of 3×3 grid with eight tiles and one empty space. AI technique used to solve it include :

- Breadth First search : Explores all possible moves level by level
- Depth First search : Explores all deeper path first
- A* Algorithm : Uses a heuristic function to find optimal path

Nikita Chhabria
Roll no. 07
DISC

04/08

output, synthetic
preference, genre data

Autolander

landing precision, passenger safety
Runway wind, attitude
Flaps, engines, landing gear
Altimeter, gyroscope, GPS

Sorting Iyer

read detection, accuracy
safety area, disorder
movement
cameras, motion sensors

shopping bot for an offline Bookstore

ability : Partially Observable
ion : Stochastic
: Sequential
Dynamic : Dynamic
ontinous : Discrete
Multi-Agent : Multi Agent

- Greedy Best First Search :- Priorities nodes based on heuristics without considering past cost

Q 4 The PEAS framework stands for Performance measure, Environment, Actuators & Sensors, it is used to define the components of an AI agent by specifying how it interacts with its environment & evaluates its success

- Taxi Driver

Performance :- Safety, speed, passenger comfort
Environment :- Roads, passengers, traffic
Actuator :- steering, acceleration, brakes
Sensors :- GPS, cameras, speedometer

- Medical Diagnosis System

P :- Diagnosis accuracy, treatment success
E :- Patients, Symptoms, Diseases
A :- Recommendations, Prescriptions
S :- Patient history, lab tests

- Music Composer AI

P :- Harmony, Originality
E :- Musical notes, instruments

A :- MIPI output, synthesis

S :- User preference, genre data

- Aircraft Autolander

P :- landing precision, passenger safety

E :- Runway wind, attitude

A :- Flaps, engines, landing gear

S :- Altimeter, gyroscope, GPS

- Robotic Sentry Gun

P :- Threat detection, accuracy

E :- security area, disorder

A :- gun movement

S :- Cameras, motion sensors

Q 5 A shopping bot for an offline Bookstore

Observability : Partially Observable

Determinism : Stochastic

Episodic : Sequential

Static / Dynamic : Dynamic

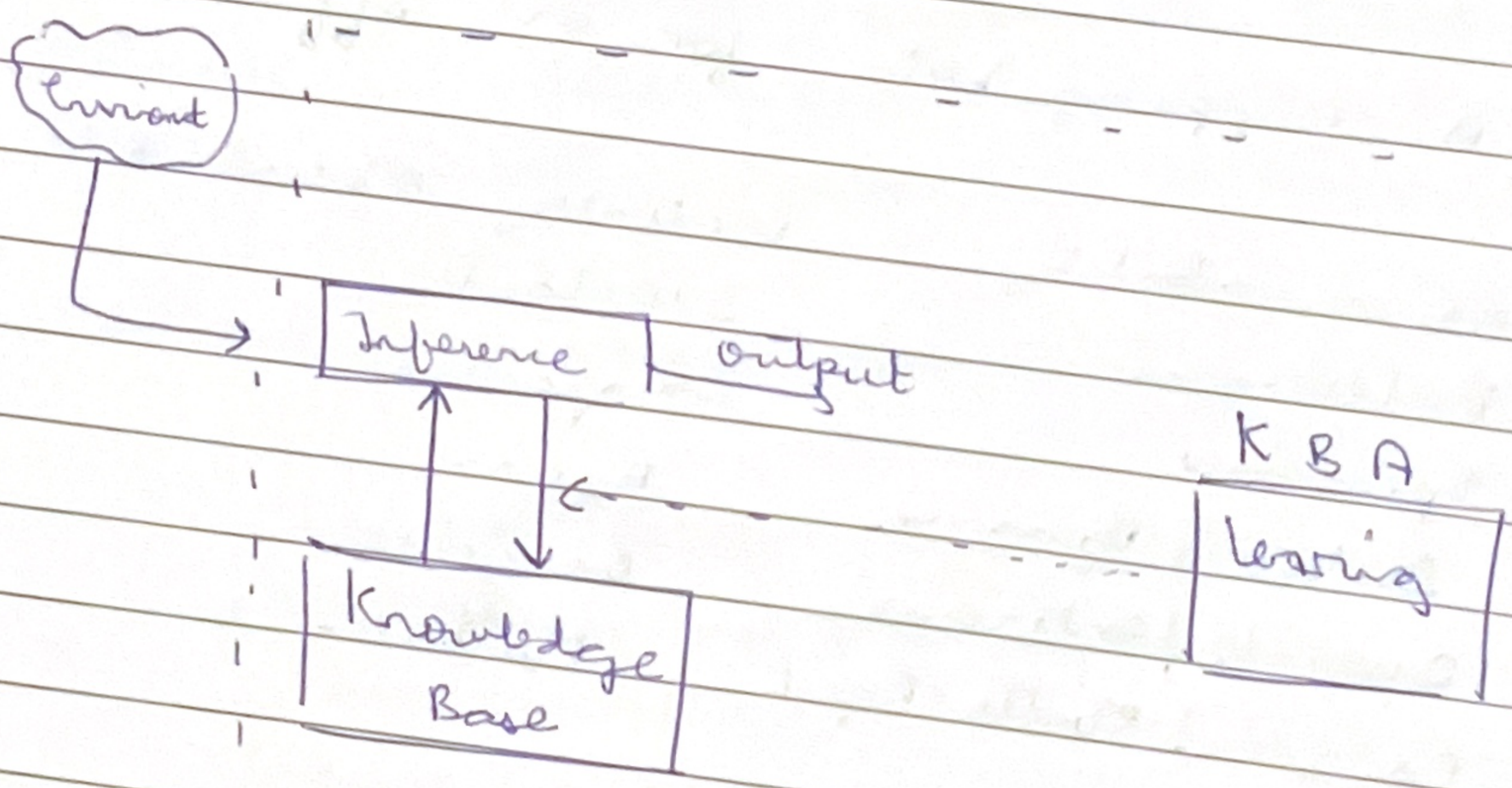
Discrete / Continuous : Discrete

Single / Multi-Agent : Multi Agent

Q1

Feature	Model Based Agent	Utility Based
Internal Model	Maintains an internal model of the world	Evaluates actions based on utility
Decision Making	Uses past state to make decision	Selects action to maximize utility
Example	Self driving cars Car Navigating through traffic	AI recommending personalized products

Q2 Architecture of knowledge based & learning Agents



A knowledge Based Agent uses stored knowledge to make decisions and reason about the environment It's architecture consists of

(1) Knowledge Base :- Stores knowledge to make decisions and reason about the environment Its architecture consists of :-

(1) Knowledge Base (KB) : Stores facts, rules and reason about in a structured format

(2) Inference Engine : Applies logical reasoning to derive conclusion from the Knowledge base

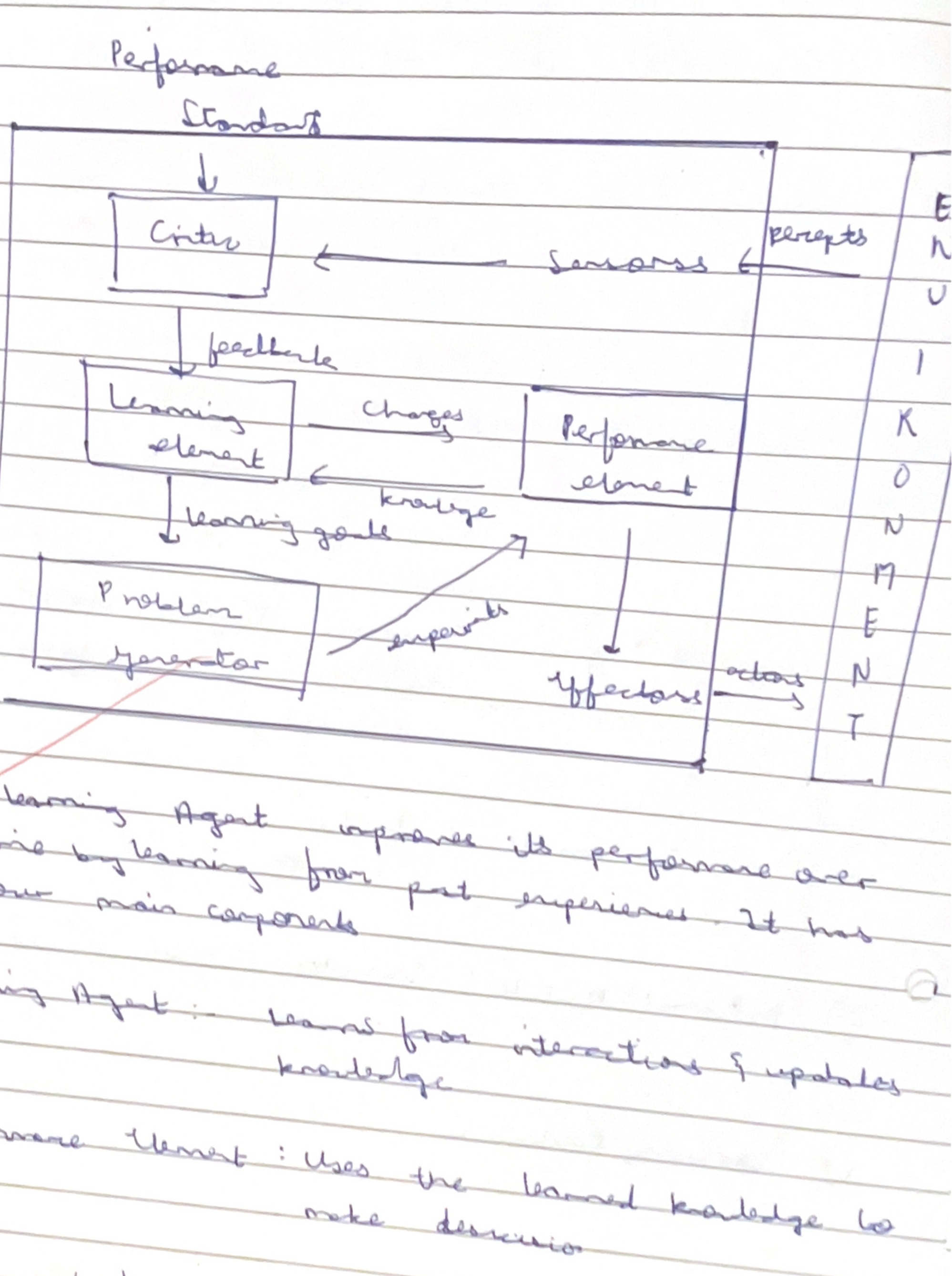
(3) Perception Module : Gather inputs from sensors or external sources

(4) Action Module : Executes action based on derived conclusions

(5) Learning Mechanism : Updates the knowledge base with new information

Example :- A medical diagnosis system that stores diseases, symptoms & applies logical rules

Q 7 Architecture of learning Agent



A learning Agent improves its performance over time by learning from past experiences. It has four main components

- learning Agent :- learns from interactions & updates knowledge
- Performance Element : Uses the learned knowledge to make decisions
- Critic : evaluates the agent's performance by comparing actions with desired outcomes

Predicate logic conversion

a) Anita travels by car if available, otherwise, travels by bus

∀ Available (car) => Travels (Anita, car)
¬ Available (car) -> Travels (Anita, Bus)

(b) Bus goes via Andheri & Goregaon

Routes (Bus, Andheri)
Route (Bus, Goregaon)

(c) Car has a puncture, so it is not available

Puncture (car) => ¬ Available (car)

Forward Reasoning

~~From (1) ¬ Available (car)~~

① Has Puncture (car) → ¬ car Available (car)

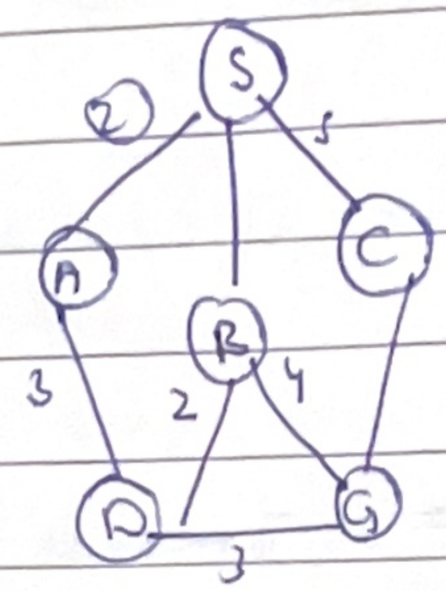
② ¬ car Available (car) → Travels (Anita, Bus)

③ Travels (Anita, Bus) [from 1 & 2]

④ Bus goes via Goregaon [from b]

⑤ Travels (Anita, Goregaon)
Yes, Anita will travel via Goregaon

q 9



S → A cost (2) A → G (cost 3)
S → B Cost 5 C → G (cost 4)
S → C cost 5
A → D cost 3

BFS Traversal from S to G

① Start at S → Queue [S]
② Expand S → Queue [A, B, C]
③ Expand B (no new nodes) → Queue [C, G]
④ Expand C → Queue [D, G, G]
⑤ Expand D → Queue [G, G]
⑥ Expand G → Goal Found

Shortest Path S to G is:

S → A → G

Total Cost : 2 + 3 = 5

q 10 Interative Deepening Search (IDS)

It combines the space efficiency of DFS & completeness of BFS by repeatedly running DLS with increasing depth limit

Advantage: guarantees finding the shortest path while using less memory than BFS

Q12 Hill Climbing Algorithm is a local search algorithm that continously moves towards the best neighboring state with a higher heuristic value, aiming to reach a global optimum

Steps

(1) Start from an intial state
(2) Evaluate neighboring states and moves to the one with the highest heuristic value
(3) Repeat until no better neighbor

Example :-

Imagine a mountain climbing scenario where a hiker moves uphill based on the steepest slope. If they reach a peak that is not the highest (global maximum), they might get stuck

Q14 Drawbacks of Hill Climbing

- Local Maximum
  The Algorithm may stop at peak that not the global optimum

- Plateau Problem :
  A flat region where all neighbouring stat have the same heuristic value, causing search to halt

- No Backtracking
  Once it moves forward, it cannot recover from a bad decision

= Limitation of Steepest - Ascent Hill Climbing

Steepest - Ascent Hill Climbing selects the best posible move in each step but has extra limitations

① Slow progress in Narrow Ridges
② More likely to get Stuck in Local Maxima
③ Ineffient in large search space

Solutions :

Random Restart
Simulated Annealing

Simulated Annealing is a probabilistic search technique that avoids getting trapped in local optima by allowing occasional bad moves

Steps :

1) Start with random solution
2) Evaluate its cost
3) Move to a neighboring solution
4) Accept it if it better, otherwise, accept it with probability decreasing over time
5) Reduce "temperature" gradually
6) Repeat with convergence

Algorithm :

Initialize temperature T, cooling rate α.
Choose an initial solution S
Repeat until $T \to 0$:
a. Select a neighbouring solution S
b. Compute cost diff $DE = cost(S') - cost(S)$
c. If $DE < 0$, move to S
d. Else, move to S' with prob $e^{(-DE/T)}$
e. Reduce T: $T = \alpha * T$

Return best solution found

Ex. Travelling Salesman Problem where a delivery truck must find shortest route

Q.14 A* is an informed search Algorithm that uses

$$f(n) = g(n) + h(n)$$

where $g(n)$ is the cost from start to node n & $h(n)$ is heuristic

Ex: Finding shortest path in a graph like Google Maps

It is widely used in pathfinding, AI & game development

Uses both $g(n)$ & $h(n)$

A* is optimal

A* is complete

Q.15 The Minimax Algorithm is used in adversarial search (eg. two-player games) to determine the optimal move by assuming both players play optimally

Maximizer (eg. X) tries to get the highest score

Minimizer (eg. O) tries to reduce the score

Algorithm

(1) Generate the game tree upto depth limit

(2) Assign heuristic values to leaf node

(3) Backpropagate values:

• Maximizer picks the maximum value

- Maximiser picks the maximum value e

The root node selects the best move based on propagated

Game tree: A minimax tree for Tic-Tac-Toe would show all possible board states, making the best move at each step

Alpha-Beta Pruning Optimises minimax by skipping unnecessary branches, reducing computation. It uses two values

Alpha (α): Best score maximiser can achieve
Beta (β): Best score Minimiser can achieve

Algorithm:

1) Perform Minimax search
2) Prune branches where:
   if Maximiser best (α) ≥ Minimiser best (β), further evaluation is skipped
   if Minimiser's best (β) ≤ Maximiser's best (α)
3) This reduces time complexity without affecting the final decision

Example: In a game tree, if a branch has already yielded a worse outcome than the best known

Q16 move, it is ignored to save time

Q17

The WUMPUS world is a grid-based AI environment where an agent explores a cave while avoiding hazards like pits & the wumpus monster

PEAS descriptor

Performance Measure: Reaching the gold safely, minimizing steps

Environment: A 4 x 4 grid with pits, gold & wumpus

Actuators: Move Forward, turn, grab, shoot, climb

Sensors: Perceive stench, breeze, glitter

Percept sequence Generation:

1) Agent start at (1, 1), sensing its surroundings

2) If stench is detected, the wumpus is nearby

3) If breeze is detected, a pit is nearby

4) The agent infers safe paths & navigates towards the gold while avoiding hazards

```
  7  5  N  D
  1  0  R  5
-------------
1  0  N  5  2
```

As E = 5 the sum of N K should be a carry

∴ N = 6 , R = 8

```
  9  5  6  D
  1  0  8  5
-------------
1  0  6  5  Y
```

Assigning the remaining value to D

∴ D = 7

```
  9  5  6  7
  1  0  8  5
-------------
1  0  6  5  2
```

Solution:-

S = 9 , M = 1

E = 5 , O = 0

N = 6 , N = 6

D = 7 , E = 5

Y = 2

Q 19 Modus Ponens is a fundamental rule of inference in propositional logic

If $P \rightarrow Q$ & P is true, then Q is also true

Example :- 1) If it rains, the ground will be wet ($P \rightarrow Q$)

1) It is raining (P is true)

3) Therefore, the ground is wet (Q is true)

Mathematically,

$P \rightarrow Q$

P

$\therefore Q$

Q 20 ① Forward Chaining is a data driven

② It starts from known facts & applies inference rules to derive new facts until the goal is reached

③ It is used in expert systems

Example :-

Fact - Sore Throat

Rule :- If sore throat → infection

New Fact :- Infection

Rule :- If infection → need antibiotics

Conclusion :- "Needs to Antibiotic

Q20 Backward Chaining

(1) It is more goal driven reference.
(2) It starts from goal & works backward to determine facts needed to prove it
(3) It is used in prolog programming

Example:-

goal : Does the patient need antibiotics?

(1) Check : Does the patient have an infection?

(2) Check : Does the patient have a sore throat?

(3) If both hold, conclude " Need antibiotic"

This reduces unnecessary computations by only exploring relevant facts