

AI

UNIT 1

1) What is AI? Explain risk and benefits of AI

Artificial Intelligence (AI)

Artificial Intelligence is the branch of computer science that focuses on creating systems capable of performing tasks that normally require human intelligence. These tasks include problem-solving, decision-making, learning, understanding natural language, recognizing patterns, and adapting to new situations. AI uses techniques like machine learning, deep learning, and natural language processing to simulate intelligent behavior.

Benefits of AI

1. Automation of Repetitive Tasks

AI can handle repetitive, time-consuming tasks with speed and accuracy, freeing humans for more creative and strategic work.

2. High Accuracy and Efficiency

AI systems can process large amounts of data quickly and make accurate predictions or decisions, often with fewer errors than humans.

3. 24/7 Availability

Unlike humans, AI systems can operate continuously without fatigue, increasing productivity.

4. Personalization

AI can tailor recommendations and services to individual users (e.g., personalized shopping, streaming suggestions).

5. Improved Decision-Making

AI analyzes huge datasets to identify patterns, trends, and predictions, helping businesses and researchers make better-informed decisions.

Risks of AI

1. Job Displacement

Automation through AI can replace certain human jobs, leading to unemployment in some sectors.

2. Bias and Discrimination

If AI is trained on biased data, it can produce unfair or discriminatory outcomes.

3. Lack of Transparency

Many AI models operate as "black boxes," making it hard to understand how decisions are made.

4. Security Risks

AI can be used for malicious purposes, such as deepfakes, cyberattacks, or autonomous weapons.

5. Overdependence

Relying too heavily on AI may reduce human skills and critical thinking abilities over time.

2) What is Intelligent Agent? Explain structure of agent and example with PEAS Property for Automatic Taxi Driving.

An agent perceives its environment through sensors

- * the complete set of inputs at a given time is called a **percept**
- * the current percept, or a sequence of percepts may influence the actions of an agent

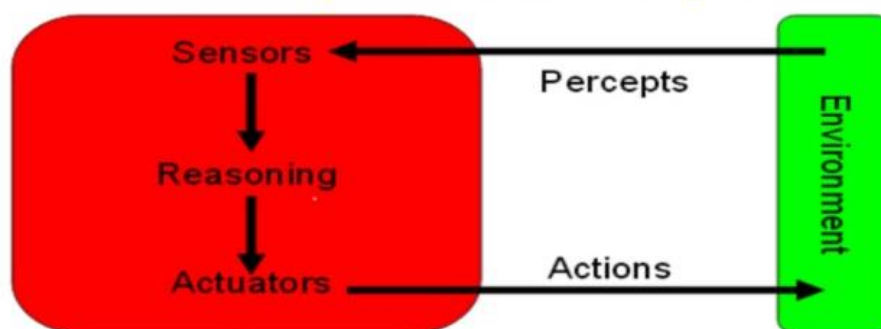
It can change the environment through actuators

- * an operation involving an actuator is called **an action**
- * actions can be grouped into action sequences

“An intelligent agent is one that is capable of **flexible autonomous** action in order to meet its design objectives, where flexible means three things:

- **reactivity**: agents are able to perceive their **environment**, and **respond in a timely fashion to changes that occur in it** in order to satisfy its design objectives;
- **pro-activeness**: intelligent agents are able to exhibit **goal-directed behavior by taking the initiative** in order to satisfy its design objectives;
- **social ability**: intelligent agents are capable of **interacting with other agents** (and possibly humans) in order to satisfy its design objectives”;

General Properties of AI Systems



This part is called an **agent**.

Agent: anything that perceives its environment through sensors and acts on that environment through actuators

PEAS Representation

PEAS = **P**erformance measure, **E**nvironment, **A**ctuators, **S**ensors.

It is used to specify the task environment of an intelligent agent.

Example – Automatic Taxi Driving Agent

1. Performance Measure (P):

- Safety of passengers and pedestrians
- Following traffic rules
- Minimizing trip time
- Maximizing passenger comfort
- Minimizing fuel consumption
- Maximizing profit

2. Environment (E):

- Roads, traffic, and weather conditions
- Other vehicles and pedestrians
- Passengers and their requests
- Road signs, signals, and lane markings

3. Actuators (A):

- Steering wheel
- Accelerator and brake
- Indicators and horn
- Gear control

4. Sensors (S):

- Cameras (to detect objects, lanes, signs)
- LIDAR / RADAR (distance sensing)
- GPS (location tracking)
- Speedometer and odometer
- Microphones (detect sirens, voice commands)



3) Explain the concept of Rationality

Concept of Rationality in AI

In Artificial Intelligence, **rationality** refers to an agent's ability to make the **best possible decision** in a given situation, based on the information it has, to achieve its goals.

A **rational agent** chooses actions that **maximize its performance measure** while considering the environment and available knowledge.

Key Points About Rationality

1. **Goal-Oriented** – Rationality is about selecting actions that help achieve the defined goals.
2. **Based on Available Information** – An agent can only be as rational as the knowledge and percepts it has at the time.
3. **Performance Measure Driven** – Rational actions aim to maximize the performance measure, not just immediate rewards.
4. **Dependent on Environment** – The “best” action can change depending on how the environment behaves.
5. **Different from Omniscience** –
 - *Omniscient agent* knows the outcome of all actions in advance (unrealistic in practice).
 - *Rational agent* chooses the best action based on what it knows **now**.

Factors Affecting Rationality

A rational decision depends on:

1. **Performance Measure** – How success is evaluated.
2. **Percept Sequence** – The history of all percepts received so far.
3. **Knowledge of Environment** – What the agent knows about how the world works.
4. **Possible Actions** – The set of actions the agent can take.
5. **Time Constraints** – Decisions may need to be made quickly.

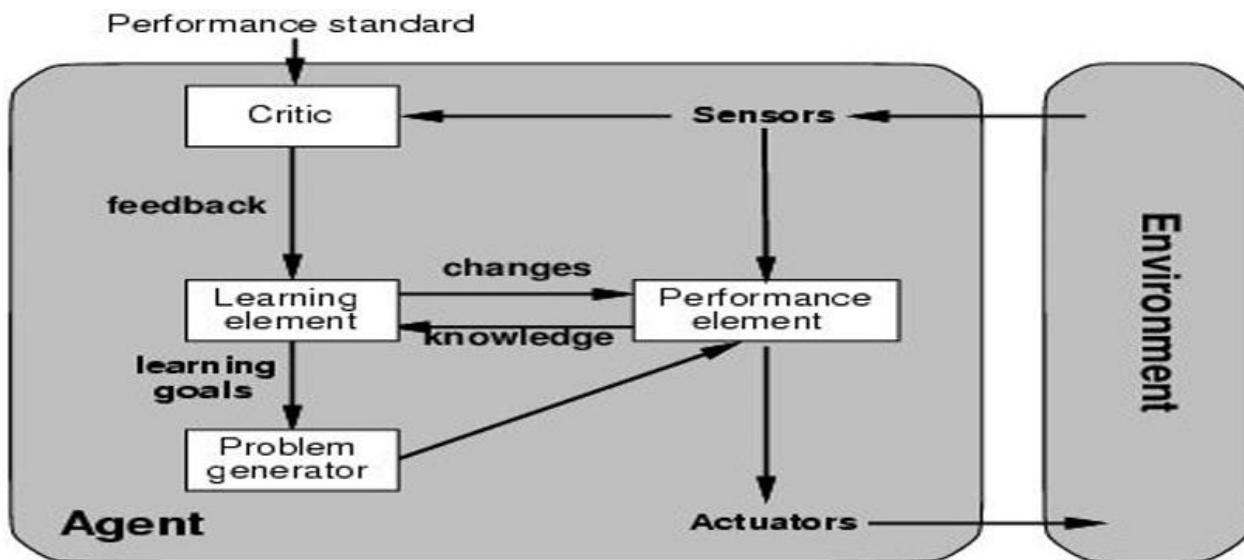
Example

For an **automatic taxi driving agent**:

- If the traffic light turns red, the **rational action** is to stop, because it:
 - Follows traffic rules (performance measure includes legality).
 - Avoids accidents (safety).
 - Avoids fines (profit and cost).

If the taxi sees a green light and an empty road, **rational action** is to proceed to save time.

4) Explain learning agents with its components.



A learning agent improves its performance over time by adapting to new experiences and data. Unlike other AI agents, which rely on predefined rules or models, learning agents continuously update their behavior based on feedback from the environment. This allows them to enhance their decision-making abilities and perform better in dynamic and uncertain situations.

* Learning agents typically consist of 4 main components:

Performance element: Makes decisions based on a knowledge base.

* **Learning element:** Adjusts and improves the agent's knowledge based on feedback and experience.

* **Critic:** Evaluates the agent's actions and provides feedback, often in the form of rewards or penalties.

* **Problem generator:** Suggests exploratory actions to help the agent discover new strategies and improve its learning.

- * For example, in [reinforcement learning](#), an agent might explore different strategies, receiving rewards for correct actions and penalties for incorrect ones. Over time, it learns which actions maximize its reward and refine its approach.
- * Learning agents are highly flexible and capable of handling complex, ever-changing environments. They are useful in applications such as autonomous driving, robotics and [virtual assistants](#) that assist human agents in [customer support](#).

5) Difference Between Simple Reflex, Model-Based Reflex, Goal-Based, and Utility-Based Agents

Aspect	Simple Reflex Agent	Model-Based Reflex Agent	Goal-Based Agent	Utility-Based Agent
1. Basis of Action	Acts only on current percept using condition-action rules.	Uses current percept + stored internal state/model of the world.	Chooses actions to achieve defined goals .	Chooses actions to maximize utility (measure of performance/happiness).
2. Memory / Internal State	✗ No memory of past percepts.	✓ Maintains memory of past events to handle partially observable environments.	✓ Can maintain model and history for planning.	✓ Maintains model and history for decision-making.
3. Environment Handling	Works only in fully observable environments.	Can handle partially observable environments.	Can handle partially observable + complex environments.	Can handle partially observable + complex + uncertain environments.
4. Goal Awareness	✗ No concept of goals; purely reactive.	✗ No goals; still reactive but smarter.	✓ Has explicit goals to achieve.	✓ Has goals and can compare between multiple possible outcomes.
5. Decision Capability	Fixed rules; no planning or reasoning.	Uses stored state to make better choices but still follows fixed rules.	Plans and searches for actions to reach goals.	Chooses the best action among goal-achieving options based on preferences/utility.
6. Example	Thermostat, automatic light switch.	Robotic vacuum that remembers where it has cleaned.	GPS navigation system planning the shortest route.	Self-driving car balancing speed, safety, and fuel efficiency.

6) Explain the Significance of PEAS in AI

What is PEAS?

PEAS stands for **Performance Measure, Environment, Actuators, and Sensors**.

It is a framework used to **define the task environment** of an intelligent agent before designing it.

Significance:

1. **Clarifies Agent's Purpose** – PEAS helps clearly define *what the agent needs to achieve* (performance measure).
2. **Identifies Working Conditions** – Specifies the *environment* in which the agent will operate, including constraints and uncertainties.
3. **Specifies Action Mechanism** – Defines the *actuators* the agent will use to affect the environment.
4. **Specifies Perception Mechanism** – Lists the *sensors* the agent will use to perceive the environment.
5. **Helps in System Design** – A well-defined PEAS description helps in selecting suitable algorithms, hardware, and software.
6. **Avoids Incomplete Design** – Ensures no component (goal, environment, sensing, action) is missed during agent development.
7. **Applicable to All AI Systems** – Works for robots, game agents, autonomous vehicles, etc., making it a universal design approach.

Example – Automatic Taxi Driving Agent

PEAS Element	Description
Performance	Safety, passenger comfort, obeying laws, minimizing time/fuel, maximizing profit
Environment	Roads, traffic, pedestrians, weather, passengers
Actuators	Steering wheel, accelerator, brake, indicators, horn
Sensors	Cameras, GPS, speedometer, LIDAR, microphones

7) Explain any four type of task environment with example.

In AI, a **task environment** is the external context in which an agent operates.

Task environments are described using properties that help decide which agent type and algorithms are needed.

1. Fully Observable vs. Partially Observable

- **Fully Observable:** The agent's sensors can access the complete state of the environment at each time.
Example: Chess game — all pieces are visible to both players.
- **Partially Observable:** Sensors provide incomplete information about the environment.
Example: Self-driving car — may not see behind a building or in fog.

2. Deterministic vs. Stochastic

- **Deterministic:** The next state of the environment is completely determined by the current state and the agent's action.
Example: Solving a Sudoku puzzle.
- **Stochastic:** The next state is not completely predictable; it has random elements.
Example: Playing poker — depends on unknown cards and opponent's moves.

3. Episodic vs. Sequential

- **Episodic:** Each action is independent of previous actions; no learning from history needed.
Example: Classifying a set of unrelated images.
- **Sequential:** Current decisions affect future actions and outcomes.
Example: Driving a car — past turns and speed affect future driving.

4. Static vs. Dynamic

- **Static:** The environment does not change while the agent is deciding.
Example: Crossword puzzle.
- **Dynamic:** The environment changes over time or while the agent is thinking.
Example: Real-time strategy game — other players move while you plan.

✓ Summary Table:

Property	Type 1	Type 2	Example	
Observability	Fully Observable	Partially Observable	Chess / Self-driving car	
Determinism	Deterministic	Stochastic	Sudoku / Poker	
Episodicity	Episodic	Sequential	Image classification / Driving	
Dynamism	Static	Dynamic	Crossword / Real-time game	

8) Explain “Simple Reflex based Agent” with the help of schematic diagram or pseudo code.

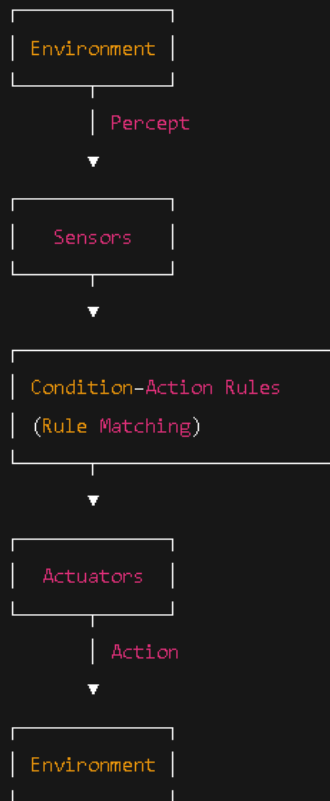
Definition:

A **Simple Reflex Agent** selects actions **only based on the current percept** and a set of condition–action rules (“if condition then action”).

It **does not** consider past percepts or future consequences.

Working:

1. Agent receives **percept** from the environment via sensors.
2. Matches the percept against **predefined condition–action rules**.
3. Executes the corresponding **action** using actuators.



Pseudo Code:

pseudo

[Copy](#) [Edit](#)

```
function SIMPLE-REFLEX-AGENT(percept):  
  for each rule in RULES:  
    if rule.condition == percept:  
      return rule.action
```

Example:

- **Thermostat Agent:**
 - **Rule 1:** IF temperature $< 20^{\circ}\text{C}$ THEN turn heater ON
 - **Rule 2:** IF temperature $\geq 20^{\circ}\text{C}$ THEN turn heater OFF

✓ Key Points:

- **Advantages:** Fast, simple to implement.
- **Limitations:** Fails in partially observable environments, no memory, cannot handle complex goals.

9) Define Artificial Intelligence. State and explain the four approaches of Artificial Intelligence?

Artificial Intelligence is the **branch of computer science** concerned with **building machines or software that can perform tasks requiring human-like intelligence** — such as reasoning, learning, problem-solving, perception, and decision-making.

Short definition (Russell & Norvig):

“Artificial Intelligence is the study of agents that receive percepts from the environment and perform actions.”

Four Approaches of Artificial Intelligence

AI can be categorized based on **thinking** vs **acting**, and **human-like** vs **rational** behavior.

Approach	Focus	Description	Example
1. Acting Humanly (Turing Test Approach)	Behavior like a human	An AI system should act in a way indistinguishable from a human during conversation or interaction.	Chatbots, voice assistants
2. Thinking Humanly (Cognitive Modeling)	Reason like a human	AI should think the way humans do, using models from cognitive science and psychology.	Cognitive architectures, ACT-R model
3. Thinking Rationally (Laws of Thought)	Reason logically	AI should use formal logic to make correct inferences and decisions.	Expert systems, theorem provers
4. Acting Rationally (Rational Agent Approach)	Take best possible actions	AI agents should choose actions that maximize performance measure given the available knowledge.	Self-driving cars, autonomous robots

Brief Explanation of Each Approach

1. Acting Humanly (Turing Test)

- Proposed by Alan Turing (1950).
- If a machine can converse and behave so well that a human judge cannot tell it apart from a real human, it passes the test.
- Requires capabilities like natural language processing, knowledge representation, reasoning, and learning.

2. Thinking Humanly (Cognitive Science)

- Models the human thought process using psychology and neuroscience.
- AI systems are built to simulate how humans actually think and solve problems.
- Example: Simulating human memory recall in problem-solving tasks.

3. Thinking Rationally (Logic-Based)

- Based on Aristotle's "laws of thought".
- AI uses formal logic to deduce conclusions from given facts.
- Limitation: Difficult to formalize all real-world knowledge into precise rules.

4. Acting Rationally (Rational Agent)

- The most widely used modern AI approach.
- An AI agent acts to achieve the best outcome or maximize expected performance.
- Works in uncertain, dynamic environments.

10) Enlist the advantages of Artificial Intelligence.

Advantages of Artificial Intelligence (AI)

1. Automation of Repetitive Tasks

- AI can handle monotonous and repetitive work without fatigue.
- Example: Assembly line robots in manufacturing.

2. 24×7 Availability

- Unlike humans, AI systems can operate continuously without breaks.
- Example: Customer support chatbots.

3. High Accuracy and Precision

- AI systems can perform tasks with minimal errors when trained properly.
- Example: Medical image diagnosis.

4. Faster Decision-Making

- AI can process and analyze large volumes of data quickly to make decisions.
- Example: Stock market prediction algorithms.

5. Ability to Work in Hazardous Environments

- AI can be deployed in places dangerous for humans.
- Example: Space exploration robots, bomb disposal robots.

6. Cost Reduction in the Long Run

- Once developed, AI reduces the need for continuous human labor in certain tasks.

7. Data Handling and Analysis

- AI can process and find patterns in massive datasets that humans cannot analyze manually.

8. Personalization

- AI tailors services or recommendations to individual preferences.
- Example: Netflix recommendations, e-commerce suggestions.

- 11) Explain the attributes used in agent design. Write PEAS description for following systems. i) A Vacuum Cleaner world system ii) Interactive English Tutor system

Attributes used in Agent Design

When designing an intelligent agent, the following attributes are considered to make it effective:

1. Performance Measure

- Criteria that evaluate the success of an agent's behavior.
- Example: For a vacuum cleaner, cleanliness of the floor.

2. Environment

- The external conditions in which the agent operates.
- Example: For a vacuum cleaner, rooms with dirt, furniture, and walls.

3. Actuators

- The mechanisms through which the agent interacts with its environment.
- Example: For a vacuum cleaner, wheels and suction mechanism.

4. Sensors

- Devices through which the agent perceives its environment.
- Example: Dirt sensors, obstacle detectors.

5. Agent Function

- The mapping from percept sequences to actions.
- Example: If dirt is detected, clean; if obstacle detected, turn.

6. Agent Architecture

- The physical or software framework that implements the agent function.
- Example: Robot body and control program for a vacuum cleaner.

PEAS Description

PEAS stands for **Performance Measure, Environment, Actuators, Sensors** — used to specify the task environment for an intelligent agent.

i) Vacuum Cleaner World System

PEAS Element	Description
Performance Measure	Amount of dirt cleaned, battery efficiency, cleaning time
Environment	Rooms, dirt, furniture, obstacles
Actuators	Wheels, suction motor, rotating brush
Sensors	Dirt detector, bump sensor, battery sensor

ii) Interactive English Tutor System

PEAS Element	Description
Performance Measure	Improvement in learner's English, user engagement, accuracy of responses
Environment	Student, learning materials, conversation context
Actuators	Text display, speech output, animations
Sensors	Keyboard input, microphone for speech recognition, camera for facial expressions

UNIT 2

- 12) Differentiate between informed and uninformed search algorithms also explain iterative deepening search in short.

1. Difference between Informed and Uninformed Search Algorithms

Aspect	Uninformed Search	Informed Search
Definition	Search algorithms that have no additional information about states beyond the problem definition.	Search algorithms that use additional domain knowledge (heuristics) to guide the search.
Knowledge Used	Only problem definition (initial state, actions, goal test).	Problem definition + heuristic information.
Efficiency	Generally slower because they explore blindly.	Faster and more efficient because they use heuristics to reduce search space.
Examples	BFS (Breadth-First Search), DFS (Depth-First Search), Uniform Cost Search.	Greedy Best-First Search, A* Search.
Optimality	Some can be optimal (e.g., BFS), but not all.	Can be optimal if heuristic is admissible and consistent (e.g., A* Search).

2. Iterative Deepening Search (IDS) – Short Explanation

- **Definition:** IDS is a search strategy that combines the space efficiency of **Depth-First Search** and the optimality of **Breadth-First Search**.
- **Working:**
 1. Perform DFS with a depth limit $d = 0$.
 2. If the goal is not found, increase depth limit by 1 ($d = d + 1$).
 3. Repeat DFS until the goal is found.
- **Advantages:**
 - Uses less memory like DFS.
 - Finds the shallowest (optimal) solution like BFS.
- **Drawback:**
 - Repeatedly explores the same nodes at shallower depths, but overhead is minimal compared to the benefit.

13) Define state-space search technique

The **state-space search technique** is a problem-solving method in which the problem is represented as a collection of **states** and **actions** that transform one state into another. The solution is found by **searching through this state space** to reach the desired **goal state** starting from the **initial state**.

- **State space** → The set of all possible states the system can be in.
- **Initial state** → The starting point of the problem.
- **Goal state** → The target state we want to reach.
- **Operators (Actions)** → The rules that describe how to move from one state to another.

Key idea:

- Model the problem as a **graph/tree** where nodes are states and edges are actions.
- Search algorithms (like BFS, DFS, A*) are applied to explore possible paths from the initial to the goal state.

Example:

In the **8-puzzle problem**, each arrangement of tiles is a state, sliding a tile is an action, and the solved arrangement is the goal state.

14) Explain problem solving agents with suitable example.

Problem-Solving Agents

A **problem-solving agent** is an intelligent agent that decides what actions to take by **formulating a problem**, **searching for a solution**, and then **executing the solution**.

It works in a **goal-based** manner — it aims to achieve a specific goal by planning before acting.

Steps of a Problem-Solving Agent

1. **Formulate the goal** – Define what needs to be achieved.
2. **Formulate the problem** – Define the initial state, actions (operators), and goal state.
3. **Search for a solution** – Apply a search algorithm to find a sequence of actions.
4. **Execute the solution** – Perform the actions in the real environment.

Components of a Problem

- **Initial state** – Where the agent starts.
- **Actions** – Possible moves or operations from a given state.
- **Transition model** – Describes the result of an action.
- **Goal test** – Checks if the current state is the goal.
- **Path cost** – Numerical cost of a solution (e.g., steps taken, time, distance).

Example – 8-Puzzle Problem

- **Initial state:** Any random configuration of tiles 1–8 and a blank space.
- **Actions:** Move blank space **up, down, left, right**.
- **Transition model:** Moving the blank space changes the positions of tiles.
- **Goal test:** Tiles arranged in order (1–8) with the blank at the end.
- **Path cost:** Number of moves made to reach the goal.

The agent will:

1. **Formulate** → Arrange tiles in correct order.
2. **Search** → Use BFS, DFS, or A* to find the shortest sequence of moves.
3. **Execute** → Make each move until the goal is reached.

- 15) Explain Min Max Tree. Solve alpha-beta tree search for following search problem. Also given minimum two advantages of alpha beta algorithm over min max algorithm.

1) What is a Min–Max tree?

A **min–max tree** is a game tree used for **two-player, zero-sum, perfect-information** games (e.g., Tic-Tac-Toe, Chess in a small horizon).

- **MAX** nodes (your turn) choose the **maximum** of their children.
- **MIN** nodes (opponent's turn) choose the **minimum** of their children.
- **Leaf nodes** carry static scores (utility values) from the evaluation function.
- The value backed up to the root is the value of the game assuming **optimal play** by both players, and the root child with that value is the **best move**.

2) Alpha–Beta pruning (how to “solve” the given tree)

Alpha (α): best value found so far on any MAX path to the node (a lower bound).

Beta (β): best value found so far on any MIN path to the node (an upper bound).

Rule:

- At a **MAX** node, prune a child as soon as its value \geq current β .
- At a **MIN** node, prune a child as soon as its value \leq current α .
- Start with $\alpha = -\infty$, $\beta = +\infty$ at the root and evaluate **left-to-right** (unless told otherwise).

3) Advantages of Alpha–Beta over plain Min–Max (any two—here are five)

1. **Same optimal result, fewer nodes expanded** (pruning doesn't change the answer).
2. **Deeper lookahead in the same time** (effective branching factor drops; best-case $\sim b^{m/2}$ vs b^m).
3. **Lower computation cost** (often exponential savings in practice).
4. **Lower memory/CPU usage** (fewer nodes generated/evaluated).
5. **Works as a drop-in to Min–Max** (just adds bounds; evaluation function unchanged).

16) Explain A* algorithm and write its pseudo code.

1. What is A* Algorithm?

A* is a **pathfinding and graph traversal algorithm**.

It is used to find the **shortest path** from a starting node to a goal node.

It's widely used in:

- Maps & GPS navigation
- Games for AI movement
- Robotics

2. How it works

A* combines:

- **Actual cost so far** from start → current node (**$g(n)$**)
- **Estimated cost** from current node → goal (**$h(n)$**) — heuristic

It evaluates each node using the formula:

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

Where:

- **$g(n)$** = cost from start to node **n**
- **$h(n)$** = estimated cost from node **n** to goal (heuristic)
- **$f(n)$** = estimated total cost of the path through **n**

The algorithm **always expands the node with the smallest $f(n)$** .

3. Steps

1. Start at the initial node, put it in an **open list** (nodes to be evaluated).
2. Pick the node from the open list with the **lowest $f(n)$** .
3. If it's the goal → path found.
4. Otherwise:
 - Move it to a **closed list** (already evaluated).
 - Check all its neighbors.
 - For each neighbor:
 - Calculate **g, h, f** .
 - If it's not in open/closed list → add it to open list.
 - If it's already in open list but with a higher cost → update it.
5. Repeat until open list is empty or goal is found.

```
A*(start, goal):
    open_set = {start}
    g_score[start] = 0
    f_score[start] = g_score[start] + heuristic(start, goal)

    while open_set is not empty:
        current = node in open_set with smallest f_score
        if current == goal:
            return path from start to goal

        remove current from open_set
        for each neighbor of current:
            temp_g = g_score[current] + distance(current, neighbor)
            if temp_g < g_score[neighbor]:
                g_score[neighbor] = temp_g
                f_score[neighbor] = temp_g + heuristic(neighbor, goal)
                add neighbor to open_set if not present

    return "No path found"
```


17) Explain hill climbing algorithm. Explain plateau, ridge, local maxima and global maxima.

- * Hill climbing algorithm is a **Heuristic search algorithm** which **continuously moves** in the direction of **increasing value** to find the **peak of the mountain or best solution to the problem**.
- * It keeps track of **one current state** and on each **iteration moves to the neighboring state with highest value** — that is, it heads in the direction that provides the **steepest ascent**.
- * In this algorithm, when it reaches a **peak value** where no neighbor has a **higher value**, then it terminates.
- * It is also called **greedy local search** as it only searches its good immediate neighbor state and not beyond that.
- * Hill Climbing is mostly used when a good heuristic is available.

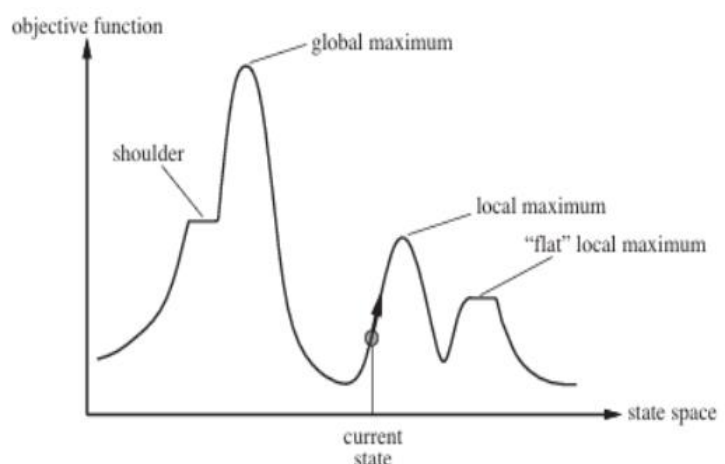
Local Maximum is a state which is better than its neighbor states, but there is also another state which is higher than it.

Global Maximum is the best possible state of state space landscape. It has the highest value of objective function.

Current state is a state in a landscape diagram where an agent is currently present.

Flat local maximum is a flat space in the landscape where all the neighbor states of current states have the same value.

Shoulder is a plateau region which has an uphill edge.



Limitations (Problems faced in Hill Climbing):

1. **Plateau** – A flat area where all neighboring states have the same value → the algorithm can get stuck without improvement.
Example: In a graph, a horizontal flat area without increase.
2. **Ridge** – A surface where the global maximum is along a narrow path; small steps miss the correct direction.
3. **Local Maxima** – A peak that is higher than nearby points but **lower than the global maximum**; algorithm stops here thinking it's the best.
4. **Global Maxima** – The highest point in the entire search space; the true best solution.

18) Define Heuristics. Explain the significance of Heuristic function in the informed search with suitable example.

Definition of Heuristics:

Heuristics are problem-specific rules or strategies used to make search algorithms more efficient by guiding them toward the most promising paths. They do not guarantee the optimal solution but help reduce search time by using extra knowledge about the problem.

Heuristic Function ($h(n)$):

- A heuristic function estimates the *cost* or *distance* from the current node **n** to the goal.
- It is used in **informed search algorithms** (e.g., Greedy Best-First Search, A* Search) to prioritize which node to explore next.

Significance in Informed Search:

1. **Guides the Search** – Chooses paths more likely to lead to the goal quickly.
2. **Reduces Search Space** – Avoids exploring irrelevant or less promising nodes.
3. **Improves Efficiency** – Finds solutions faster compared to blind search (e.g., BFS, DFS).
4. **Influences Optimality** – With a good heuristic (admissible & consistent), optimal solutions can be guaranteed in algorithms like A*.

Example:

Suppose we want to find the shortest path from city **A** to city **G** on a map.

- **$h(n)$** = straight-line distance from current city **n** to **G**.
- If at city **B**, and straight-line distance to G is 200 km, then **$h(B) = 200$** .
- The search algorithm will prefer expanding cities with smaller **$h(n)$** values, as they are closer to the goal.

Formula in A* Search:

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

where:

- **$g(n)$** = cost from start to node **n**
- **$h(n)$** = estimated cost from **n** to goal
- **$f(n)$** = estimated total cost of path through **n**

19) Define problem? Write & explain the five components of Well-defined problem.

Problem – Definition:

In Artificial Intelligence (AI), a **problem** is a situation where an *agent* is in a given **initial state** and must perform actions to reach a desired **goal state**. The solution to a problem is a sequence of actions that transforms the initial state into the goal state.

Five Components of a Well-Defined Problem

A problem is considered **well-defined** when it is clearly specified using the following five components:

1. Initial State

- The state in which the agent starts before taking any action.
- It is the starting point for the search process.
- **Example:** In the 8-puzzle problem, the initial arrangement of tiles.

2. State Space

- The set of all possible states reachable from the initial state by applying a sequence of actions.
- It represents the entire search area.
- **Example:** All possible configurations of the 8-puzzle tiles.

3. Actions (Operators)

- The set of legal moves or operations the agent can perform to change from one state to another.
- **Example:** In 8-puzzle, moving the blank tile up, down, left, or right.

4. Goal Test

- A method to check whether the current state is the desired goal state.
- **Example:** In 8-puzzle, the goal test checks whether all tiles are in the correct order.

5. Path Cost Function

- A function that assigns a numeric cost to each path, representing the total cost of moving from the initial state to a given state.
- It helps in finding the optimal (least costly) solution.
- **Example:** In route planning, the path cost could be the total distance traveled.

Example – 8-Puzzle Problem

- **Initial State:** Random arrangement of tiles.
- **State Space:** All possible tile arrangements.
- **Actions:** Move blank tile up, down, left, or right.
- **Goal Test:** Tiles are in correct order.
- **Path Cost:** Number of moves made.

20) Explain different search strategies.

In **Artificial Intelligence (AI)**, a **search strategy** is the method used to explore the problem space to find a solution.

Search strategies are generally divided into two main categories:

1. Uninformed Search Strategies (Blind Search)

These strategies **do not have any additional information** about the goal beyond what is provided in the problem definition. They explore the search space blindly.

Types:

1. Breadth-First Search (BFS)

- Explores all nodes at the current depth before going deeper.
- Complete & optimal (for equal step costs).
- Example: Searching level-by-level in a tree.

2. Depth-First Search (DFS)

- Explores as far as possible down one branch before backtracking.
- Less memory, but not always complete and may get stuck in infinite paths.

3. Depth-Limited Search (DLS)

- DFS with a fixed depth limit.
- Prevents infinite loops but may miss solutions beyond the limit.

4. Iterative Deepening Search (IDS)

- Repeatedly applies DLS with increasing depth limits.
- Combines BFS completeness with DFS space efficiency.

5. Uniform Cost Search (UCS)

- Expands the node with the lowest path cost.
- Guarantees optimal solution for varying step costs.

2. Informed Search Strategies (Heuristic Search)

These strategies use **extra knowledge** (heuristics) to make searching more efficient.

Types:

1. Best-First Search

- Expands the node that appears best according to a heuristic function.

2. Greedy Best-First Search

- Chooses the node closest to the goal according to a heuristic ($h(n)$).
- Fast but not always optimal.

3. A* Search

- Uses both path cost ($g(n)$) and heuristic ($h(n)$):

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

- Optimal if $h(n)$ is admissible (never overestimates).

- 21) Explain Depth-limited search and Iterative deepening depth-first search. Compare both search strategies w.r.t. Completeness, Optimality, Space Complexity and Time Complexity.

1. Depth-Limited Search (DLS)

- **Definition:**

Depth-Limited Search is a variation of Depth-First Search (DFS) where the search is restricted to a pre-defined depth limit l .

This helps avoid infinite loops in cases where the search tree is infinite.

- **Working:**

1. Start from the root node.
2. Explore nodes depth-first, but do not go beyond the given depth limit.
3. If the depth limit is reached, stop exploring that path.

- **Example:**

Suppose a depth limit of $l = 3$. The search will explore all paths until depth 3 only, ignoring deeper nodes.

2. Iterative Deepening Depth-First Search (IDDFS)

- **Definition:**

IDDFS combines the **space efficiency of DFS** with the **completeness of BFS**.

It repeatedly performs DLS with increasing depth limits until the goal is found.


- **Working:**

1. Perform Depth-Limited Search with limit $l = 0$.
2. If the goal is not found, increase l to 1, then 2, and so on.
3. Stop when the goal is found.

- **Example:**

First search at depth 0, then depth 1, depth 2... until the goal is reached.


3. Comparison Table

Feature	Depth-Limited Search (DLS)	Iterative Deepening DFS (IDDFS)	
Completeness	✗ Not complete (may miss goal if depth limit < depth of goal)	✓ Complete if branching factor is finite	
Optimality	✗ Not optimal (depends on limit)	✓ Optimal if step cost is uniform	
Space Complexity	$O(b \times l)$ where b = branching factor, l = depth limit	$O(b \times d)$ where d = depth of shallowest goal	
Time Complexity	$O(b^l)$	$O(b^d)$ (repeated nodes, but still efficient)	

- ✓ **Key Insight:**

- **Use DLS** when you know the exact depth of the goal.
- **Use IDDFS** when you don't know the goal depth but want completeness & low space usage.

22) Compare Depth First and Best First Search methods.

Aspect	Depth First Search (DFS)	Best First Search (BestFS) 
Approach	Explores as far as possible along each branch before backtracking.	Expands the node that appears most promising based on a heuristic function.
Data Structure Used	Stack (LIFO) – can be implemented recursively.	Priority Queue – ordered by heuristic value.
Heuristic Used	No heuristic; it's uninformed search.	Uses a heuristic function $h(n)$ to guide the search.
Completeness	Not complete (may go into infinite paths in infinite state space).	Complete if the branching factor is finite and heuristic leads toward goal.
Optimality	Not optimal (does not guarantee shortest path).	Not necessarily optimal unless combined with other techniques (like A*).
Time Complexity	$O(b^m)$ — b = branching factor, m = maximum depth.	$O(b^m)$ in worst case but often faster due to heuristic guidance.
Space Complexity	$O(b*m)$ — stores only the current path.	$O(b^m)$ — stores all generated nodes in the priority queue.
When to Use	When memory is limited and solution depth is known.	When a good heuristic is available to speed up search.
Example Use Case	Puzzle solving with unknown goal distance.	Pathfinding in maps when distance estimates are available.

23) What is an Optimization problem?

An **Optimization Problem** is a type of problem where the goal is to find the *best possible solution* from a set of feasible solutions, based on certain criteria or constraints.

In other words, instead of just finding **any** solution, you aim for the **most optimal one** — for example, the shortest route, the cheapest cost, or the maximum profit.

Key Points

- **Objective function:** The function you want to *maximize* or *minimize*.
- **Constraints:** Conditions that the solution must satisfy.
- **Search space:** All possible solutions.
- **Optimal solution:** The solution that best satisfies the objective function.

Examples

1. Traveling Salesman Problem (TSP)

Find the shortest possible route that visits all cities exactly once and returns to the start.

2. Job Scheduling

Assign jobs to machines to minimize total completion time.

In AI

Optimization problems often use **heuristic** or **metaheuristic** algorithms like:

- A* Search
- Genetic Algorithms
- Simulated Annealing

24) Write a brief note on foundation and history of AI. (UNIT - I)

Foundation and History of AI

The field of **Artificial Intelligence (AI)** aims to create machines that can perform tasks that normally require human intelligence, such as problem-solving, learning, reasoning, and understanding language.

Foundation of AI

AI is built upon several disciplines:

- **Mathematics** – for logic, algorithms, probability, and optimization.
- **Computer Science** – for programming, data structures, and computational models.
- **Psychology & Cognitive Science** – to understand human thinking and learning processes.
- **Neuroscience** – for brain-inspired computing.
- **Linguistics** – for natural language processing.

History of AI (Brief Timeline)

- **1943** – *McCulloch & Pitts* created a model of artificial neurons.
- **1950** – *Alan Turing* proposed the **Turing Test** to measure machine intelligence.
- **1956** – The term "**Artificial Intelligence**" was coined at the **Dartmouth Conference** (John McCarthy, Marvin Minsky, Allen Newell, Herbert Simon).
- **1960s–1970s** – Development of early AI programs like ELIZA (natural language) and Shakey (robot).
- **1980s** – Rise of **expert systems** (e.g., MYCIN).
- **1997** – IBM's **Deep Blue** defeated chess champion Garry Kasparov.
- **2010s** – Breakthroughs in **deep learning** (e.g., ImageNet, speech recognition, self-driving cars).
- **Present** – AI is integrated into everyday technology: virtual assistants, recommendation systems, medical diagnosis, autonomous vehicles.

All The Best!!

- Karan Salunkhe 😊