Times asked: 6 times
5 times
4 times
3 times
2 times
1 time
# indicates 5-mark question

AI Question Bank

1.Introduction to AI

1. Describe four categories of Artificial Intelligence. #

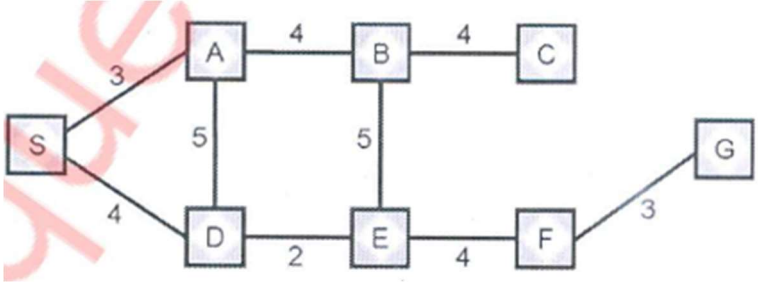
2. Intelligent Agents

1. Explain PEAS descriptor. Also state PEAS description for given object. PYQs:
   i. Vacuum cleaner robot
   ii. Automobile Driver agent
   iii. Part picking robot
   iv. Medical diagnosis system
   v. Online English tutor.
2. Explain Problem formulation, also give the initial state, goal test, successor function, and cost function for the following. Choose the formulation that is precise enough to be implemented. PYQs:
   i. Problem Statement: Autonomous Taxi driver
   ii. Wumpus world problem
   iii. Problem statement: A 3-foot-tall monkey is in a room where some bananas are suspended from the 8-foot-tall ceiling. He would like to get bananas. The room contains two stackable, movable, climbable 3-foot-high crates.
   iv. Formulate the 8-puzzle problem. #
3. List down all agent types. Explain each with block diagram.
4. Explain various properties of Task environment with suitable examples.

3. Problem Solving

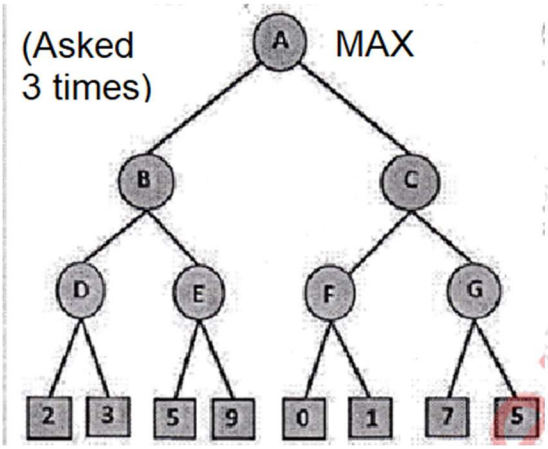
1. Explain hill climbing algorithm and problems that occur in hill climbing algorithm along with solutions.
2. Apply A* algorithm on a given graph.

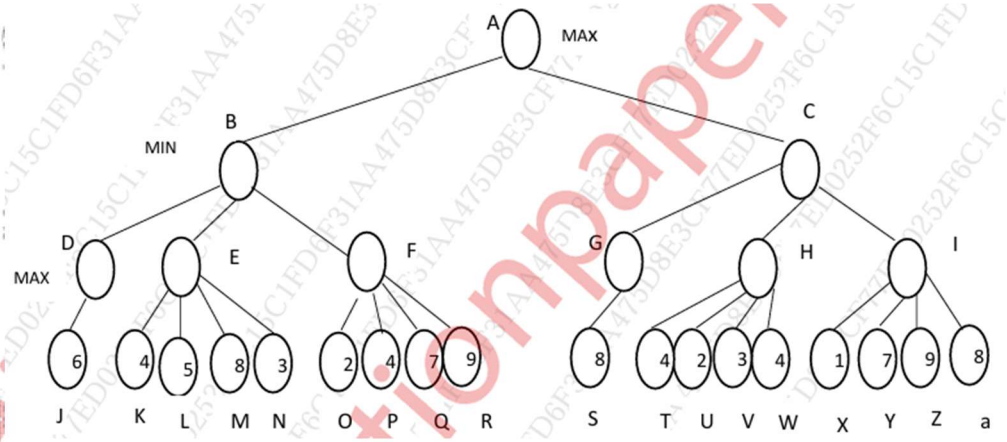
PYQ: Heuristic values are h(S)=15, h(A)=14, h(D)=12, h(B)=10, h(E)=10, h(C)=8, h(F)=10, h(G)=0. S is the start node and G is the goal node.



(Asked twice)

3. Apply alpha beta pruning on given graph, PYQs:





4. Explain alpha beta search in detail with example.
5. Differentiate between Informed search and Uninformed search algorithms. #
6. Explain the Depth Limited search and Depth first iterative deepening search.
7. Explain Simulated annealing with suitable example.

4. Knowledge and Reasoning

1. Explain forward-chaining and backward-chaining algorithm in detail with suitable example.
2. Write a detailed note on Wumpus world environment.
3. Solve given problem using Resolution:

i. Consider the following statements: (asked twice)

a) All people who are graduating are happy
b) All happy people smile
c) Someone is graduating

Represent above statements in FOL, Convert each to CNF, Prove that "someone is smiling" using resolution. Draw the resolution tree.

ii. What actions would you take to prove "Some who are intelligent can't read" using propositional logic:

a) Whoever can read is literate
b) Dolphins are not literate
c) Some dolphins are intelligent

iii. Consider the following facts:

a) Steve only likes easy courses.
b) Science courses are hard.
c) All the courses in the basket _ weaving department are easy.
d) BK301 is a basket _ weaving course.

Find by resolution that "What course would Steve like?"

5. Planning and Learning

1. What is planning in AI? Explain Partial-order planning with suitable example.
2. Problem on planning, PYQs:
   - **i.** Design a planning problem using STRIP for Air cargo transport. It involves loading and unloading cargo onto and off of planes and flying it from place.
     Initial state: At SFO airport, Cargo1, Plane1 and at JFK airport, Cargo2, Plane2 is present.
     Goal state: At SFO airport Cargo2 and at JFK airport Cargo1 is present.
   - **ii.** Consider problem of changing a flat tire. The goal is to have a good spare tire properly mounted on to the car's axle, where the initial state has a flat tire on the axle and a good spare tire in the trunk. Give the ADL description for the problem and also discuss the solution.
3. Explain the concept of PAC learning.
4. Explain Reinforcement learning in detail.
5. Explain hierarchical planning in detail. #

6. AI Applications

1. Explain different applications of AI in Robotics, Healthcare, Retail and Banking.
2. Write detailed note on: Language models of Natural Language Processing.
3. Give types of parsing and generate the parse tree for the sentence "The cat ate the fish." #

	1	2	3	4	5	6
2024 Dec	5	20	30	30	30	20
2024 May	0	20	35	45	20	10
2023 Dec	5	20	30	40	20	20
2023 May	5	20	55	25	10	10
2022 Dec	5	25	40	20	30	20
Last 5 Avg	**5**	**20**	**35-40**	**30-40**	**20-30**	**10-20**
***2022 May**	0	10	35	25	15	10
Total	20	115	225	185	125	90

Asked once:

# indicates 5-mark question

1.Introduction to AI

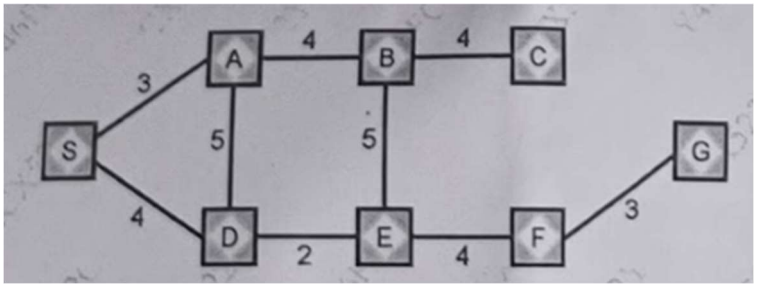
1. What do you mean by Total Turing test. #

2. Intelligent Agents

1. What do you mean by state space representation? Explain with example the necessity of it. #

3. Problem Solving

1. Apply greedy best-first search. At each iteration, each node is expanded using evaluation function f(n) = h(n). h(S)=10, h(A)=10, h(D)=8, h(B)=6, h(E)=6.5, h(C)=4, h(F)=3, h(G)=0. S is the start state and G is goal state.



2. What is Game playing algorithm? Draw a game tree for Tic-Tac-Toe problem.
3. What do you understand by Min Max Search? Explain in detail with example. #
4. What do you understand by informed and uninformed search methods? Explain in detail with example.
5. Explain the concept of genetic programming.

4. Knowledge and Reasoning

1. Explain different quantifiers with example. #
2. What do you mean by Resolution? Also discuss the steps in Resolution.
3. Define Belief network. Describe the steps of constructing belief network with an example.
4. Explain various methods of knowledge representation.
5. Compare and contrast propositional logic and first order logic. #
6. Write a short note on conditional probability and its role in AI. #

5. Planning and Learning

1. Explain the concept of Conditional order planning. #
2. Compare the importance of Partial order planning over Total order planning. #
3. What data is used to evaluate award and punishment of robot navigation. #
4. Explain the concept of Supervised learning. #

6. AI Applications

None

AI Question Bank

multiple times asked questions highlighted
question asked once with red font
# indicates 5-mark question

1.Introduction to AI

1. Describe four categories of Artificial Intelligence. #

1. Acting Humanly (The Turing Test Approach)

- **Definition**: AI systems that try to behave like a human.
- **Goal**: Make a machine that acts like a person and passes the Turing Test, meaning a human can't tell it apart from another human in conversation.
- **Example**: Chatbots like ChatGPT.

2. Thinking Humanly (The Cognitive Modelling Approach)

- **Definition**: AI systems that think like humans, mimicking human cognitive processes.
- **Goal**: Understand how humans think and replicate it in machines.
- **Example**: Cognitive architectures like ACT-R and Soar.

3. Thinking Rationally (The Laws of Thought Approach)

- **Definition**: AI systems that use logic and reasoning to reach conclusions.
- **Goal**: Develop machines that can think logically and follow correct reasoning paths.
- **Example**: Expert systems.

4. Acting Rationally (The Rational Agent Approach)

- **Definition**: AI systems that take rational actions to achieve the best outcome.
- **Goal**: Machines that act intelligently and optimally in any given situation.
- **Example**: Autonomous robots, self-driving cars.

2. What do you mean by Total Turing test.

The Total Turing Test is an advanced version of the original Turing Test, proposed by Alan Turing in 1950 to determine whether a machine can exhibit human-like intelligence.

Definition:

The Total Turing Test evaluates a machine's ability to exhibit intelligent behaviour, not only through textual conversation but also by demonstrating:

1. **Perceptual abilities** (vision, speech recognition, etc.)
2. **Physical interaction** with the environment (using robotics)

Capabilities of Total Turing test:

Natural Language Processing (NLP): To communicate fluently in human language

Knowledge Representation: To store and retrieve general world knowledge

Automated Reasoning: To use stored knowledge for problem-solving and decision-making

Machine Learning: To improve performance based on experience

2. Intelligent Agents

3. Explain PEAS descriptor. Also state PEAS description for given object. PYQs:
    i. Vacuum cleaner robot
    ii. Automobile Driver agent
    iii. Part picking robot
    iv. Medical diagnosis system
    v. Online English tutor.

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

Performance Measure: It is the objective function used to evaluate the agent's performance.

Environment: It refers to the real-world setting in which the agent operates and takes actions.

Actuators: These are the tools, devices, or mechanisms the agent uses to perform actions in the environment. They serve as the agent's output.

Sensors: These are the tools, devices, or mechanisms the agent uses to perceive and capture the state of the environment. They serve as the agent's input.

i. Vacuum Cleaner Robot

- **Performance Measure**:
    1. Cleanliness of the floor
    2. Energy efficiency (battery usage)
- **Environment**:
    1. Rooms with furniture
    2. Dirt and obstacles on floor
- **Actuators**:
    1. Wheels for movement
    2. Vacuum suction motor
- **Sensors**:
    1. Dirt sensor
    2. Obstacle detection sensors (IR/ultrasonic)

ii. Automobile Driver Agent

- **Performance Measure**:
    1. Safe and legal driving
    2. Passenger comfort and fuel efficiency
- **Environment**:
    1. Roads and traffic
    2. Pedestrians and other vehicles

- **Actuators**:
    1. Steering wheel, accelerator, brake
    2. Gear system and indicators
- **Sensors**:
    1. Cameras, sonar, GPS
    2. Speedometer, lidar

iii. Part Picking Robot

- **Performance Measure**:
    1. Accuracy of part placement
    2. Speed of operation
- **Environment**:
    1. Conveyor belt with parts
    2. Storage bins or sorting trays
- **Actuators**:
    1. Robotic arm
    2. Gripper or hand
- **Sensors**:
    1. Camera for part recognition
    2. Joint angle or position sensors

iv. Medical Diagnosis System

- **Performance Measure**:
    1. Accuracy of diagnosis
    2. Minimizing cost
- **Environment**:
    1. Patient data and medical history
    2. Hospital or clinical setting
- **Actuators**:
    1. Display diagnosis and recommendations
    2. Suggest tests or refer to specialists
- **Sensors**:
    1. Keyboard or digital input for symptoms
    2. Medical report inputs/test results

v. Online English Tutor

- **Performance Measure**:
    1. Student performance improvement
    2. Engagement and completion rates
- **Environment**:
    1. Online learning platform
    2. Interactions with students
- **Actuators**:
    1. Display lessons and feedback
    2. Provide suggestions and corrections
- **Sensors**:
    1. Keyboard/microphone input from students
    2. Test answers or learning activity tracking

4. Explain Problem formulation, also give the initial state, goal test, successor function, and cost function for the following. Choose the formulation that is precise enough to be implemented. PYQs:
    i. Problem Statement: Autonomous Taxi driver
    ii. Wumpus world problem
    iii. Problem statement: A 3-foot-tall monkey is in a room where some bananas are suspended from the 8-foot-tall ceiling. He would like to get bananas. The room contains two stackable, movable, climbable 3-foot-high crates.
    iv. Formulate the 8-puzzle problem. #

Problem formulation is the process of defining a problem clearly so that an AI agent can solve it using search algorithms. It involves identifying:

1. **Initial State** – Where the agent starts.
2. **Actions / Successor Function** – What the agent can do.
3. **Goal Test** – How to know if the problem is solved.
4. **Path Cost / Cost Function** – The cost of a path (e.g., time, distance, fuel, etc.)

i. Problem Statement: *Autonomous Taxi Driver*

A self-driving taxi agent must pick up passengers from one location and drop them at another while avoiding traffic and obeying rules.

1. Initial State:

The starting position of the taxi, location of the passenger, and destination.

e.g.; Taxi at (2,3), passenger at (1,1), destination at (5,5)

2. Successor Function (Actions):

The set of legal actions the taxi can take:

- Move North, South, East, West (if road exists)
- Pick up passenger
- Drop off passenger

Each action transitions the state accordingly.

3. Goal Test:

The passenger has been successfully dropped at the destination.

Taxi at destination AND passenger delivered

4. Path Cost Function:

Total cost could include:

- Distance travelled
- Fuel consumed
- Time taken
- Number of illegal moves (penalized)

Cost = distance + traffic penalties + fuel consumption

ii. Wumpus world problem

The Wumpus World is a classic AI problem used to illustrate logical reasoning and knowledge representation. It is a 4x4 grid-based environment with the following features:

- The agent starts in cell (1,1) facing right.
- There may be pits (deadly), a Wumpus (a monster), and gold.
- The agent can move forward, turn left/right, grab gold, shoot an arrow, or climb out.

Initial State:

- Agent is at (1,1), facing right.
- Arrow available.
- Gold not yet collected.

(state = (1,1), direction = right, has_gold = false, arrow = true)

Successor Function (Actions):

- **MoveForward** – move one cell in the direction the agent is facing.
- **TurnLeft / TurnRight** – changes the agent's direction.
- **Grab** – pick up gold if it's in the current cell.
- **Shoot** – shoot arrow in current direction (kills Wumpus if hit).
- **Climb** – climb out of cave (only from (1,1)).

Goal Test:

- Agent has collected the gold and climbed out of the cave alive.

has_gold = true AND agent_alive = true AND position = (1,1) AND climbed_out = true

Path Cost Function:

Total cost could include:

- Distance travelled
- Fuel consumed
- Time taken
- Number of illegal moves (penalized)

Cost = distance + traffic penalties + fuel consumption

iii. Problem Statement: Monkey and Bananas Problem

A 3-foot-tall monkey wants to grab bananas hanging from the 8-foot ceiling. There are two 3-foot-tall crates in the room, which are movable, climbable, and stackable. The monkey must figure out how to use the crates to reach the bananas.

Initial State:

- Monkey is on the floor at some location.
- Two crates are on the floor at specific locations.
- Crates are not stacked.
- Monkey is not holding anything or standing on a crate.

(monkey_position = A, crate1_position = B, crate2_position = C, crates_stacked = false, monkey_on_crate = false, has_bananas = false)

Successor Function (Actions):

- **Move(x):** Move monkey to location x.
- **Push(crate, x):** Push crate to location x.
- **Climb(crate):** Climb on top of crate or stack of crates.
- **Stack(crate1, crate2):** Stack one crate on another (if in same location).
- **Unstack:** Remove crate from on top of another.
- **Grab Bananas:** Grab bananas if within reach (i.e., 3ft + 3ft + monkey = 9ft ≥ 8ft ceiling).

Goal Test:

- Monkey has successfully grabbed the bananas.

has_bananas = true

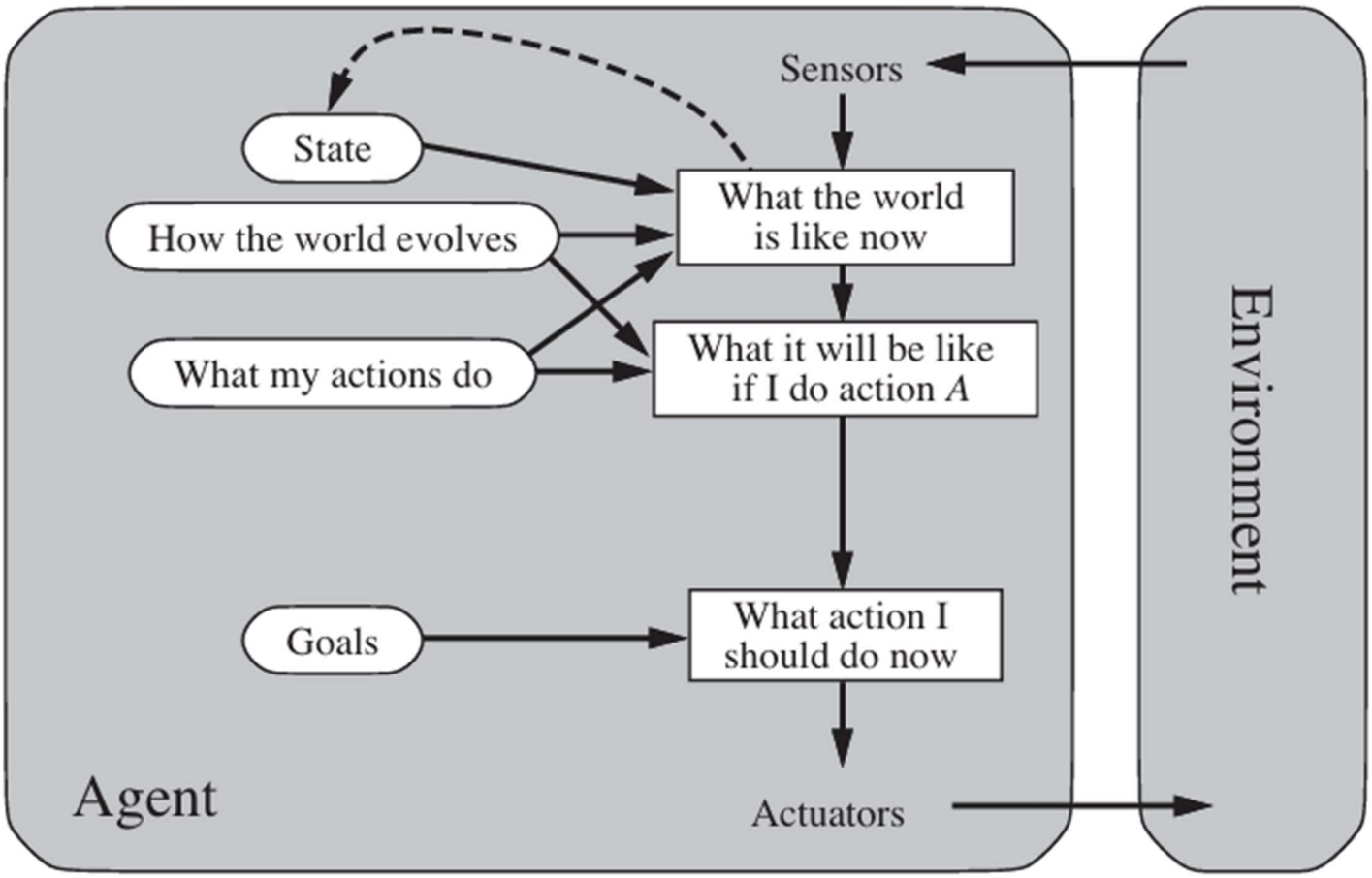
Path Cost Function:

- Each action may have a cost of 1.
- Total cost = number of actions taken.

5. List down all agent types. Explain each with block diagram.

(This question was asked once, but Goal-based, Utility-based, and Learning agents were each asked for 5 marks in different papers as well.)

Goal based agent



A Goal-Based Agent makes decisions by considering future outcomes and choosing actions that achieve a specific goal. Unlike reflex agents, it doesn't just react — it reasons and plans.

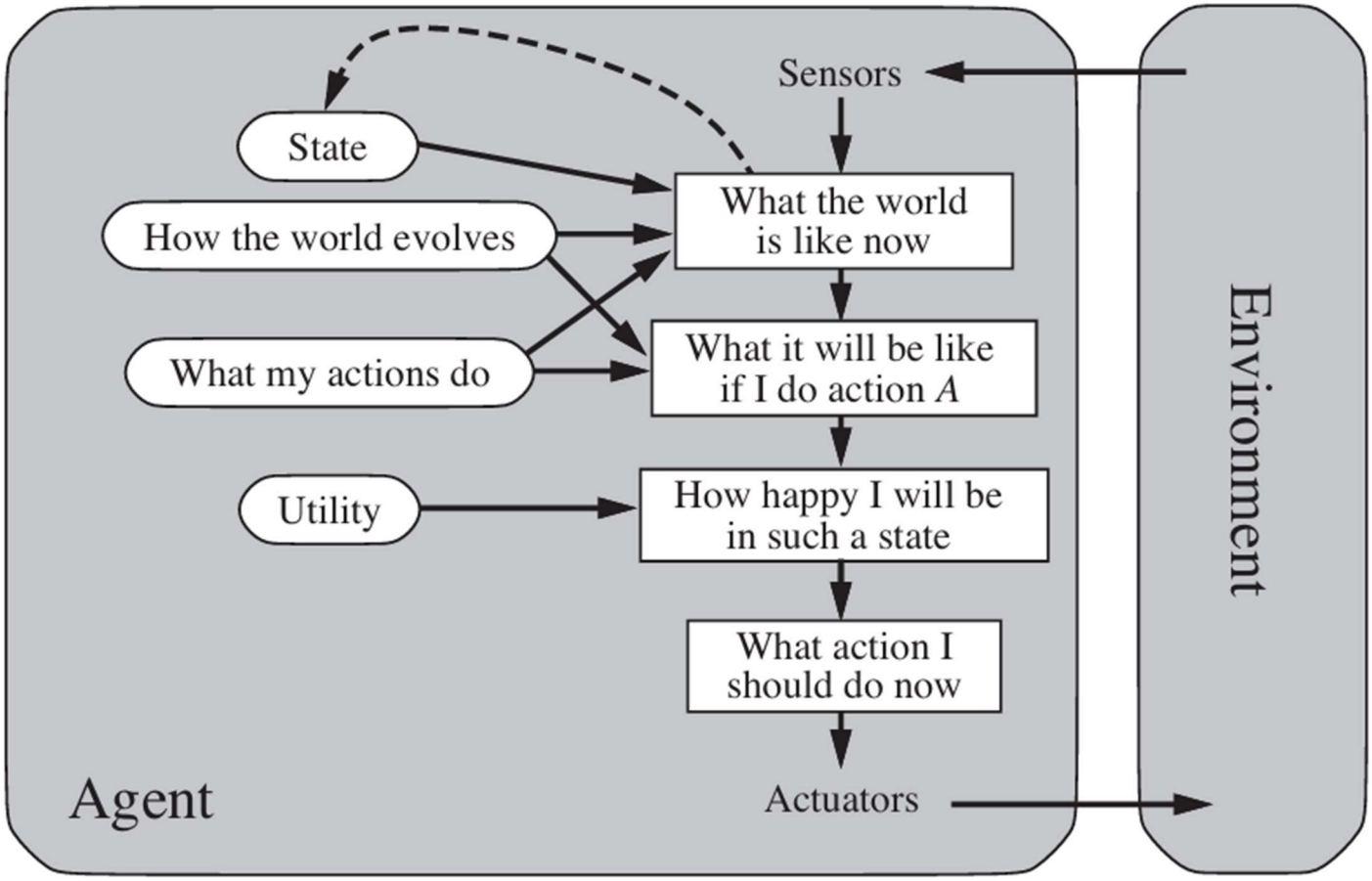
Key Characteristics:

- Uses goal information to decide what action to take.
- Can perform search or planning to find a sequence of actions.
- More flexible and intelligent than reflex agents.
- Suitable for dynamic and complex environments.

Example:

An automated delivery robot planning the shortest path to deliver a package to a specific address.

Utility based agent



A Utility-Based Agent enhances goal-based decision-making by choosing actions that maximize overall happiness, efficiency, or performance, using a utility function.

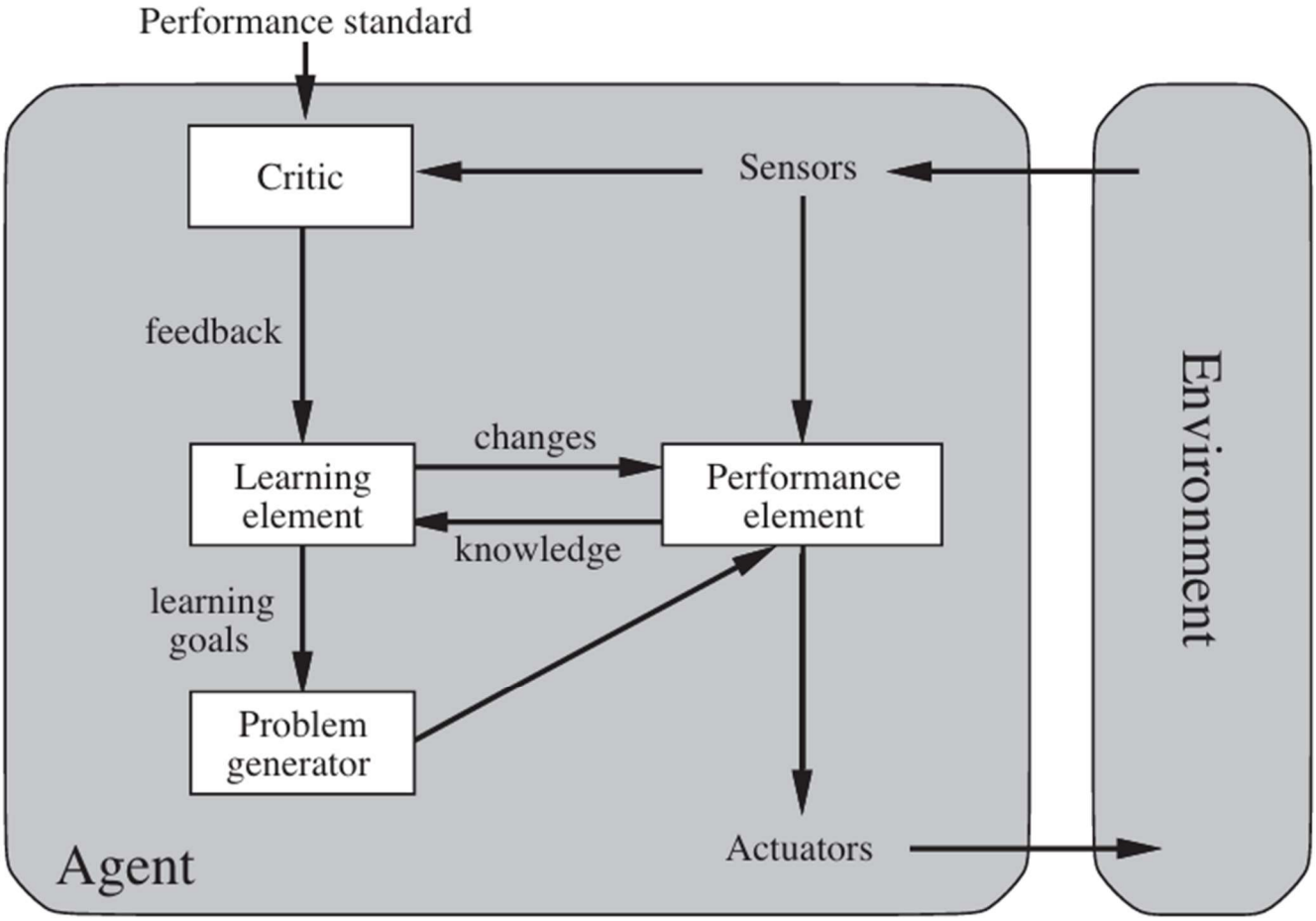
Key Characteristics:

- Uses a utility function to measure the "goodness" of outcomes.
- Chooses among multiple possible goals or actions based on expected utility.
- Handles conflicting goals, risk, and uncertainty better than goal-based agents.

Example:

A self-driving car selecting a route that’s not just shortest, but also safest and most comfortable for the passenger.

Learning agent:



A Learning Agent is an intelligent agent that can improve its performance over time by learning from experience and feedback from the environment.
It is especially useful in complex environments where hardcoding all behaviours isn't practical.

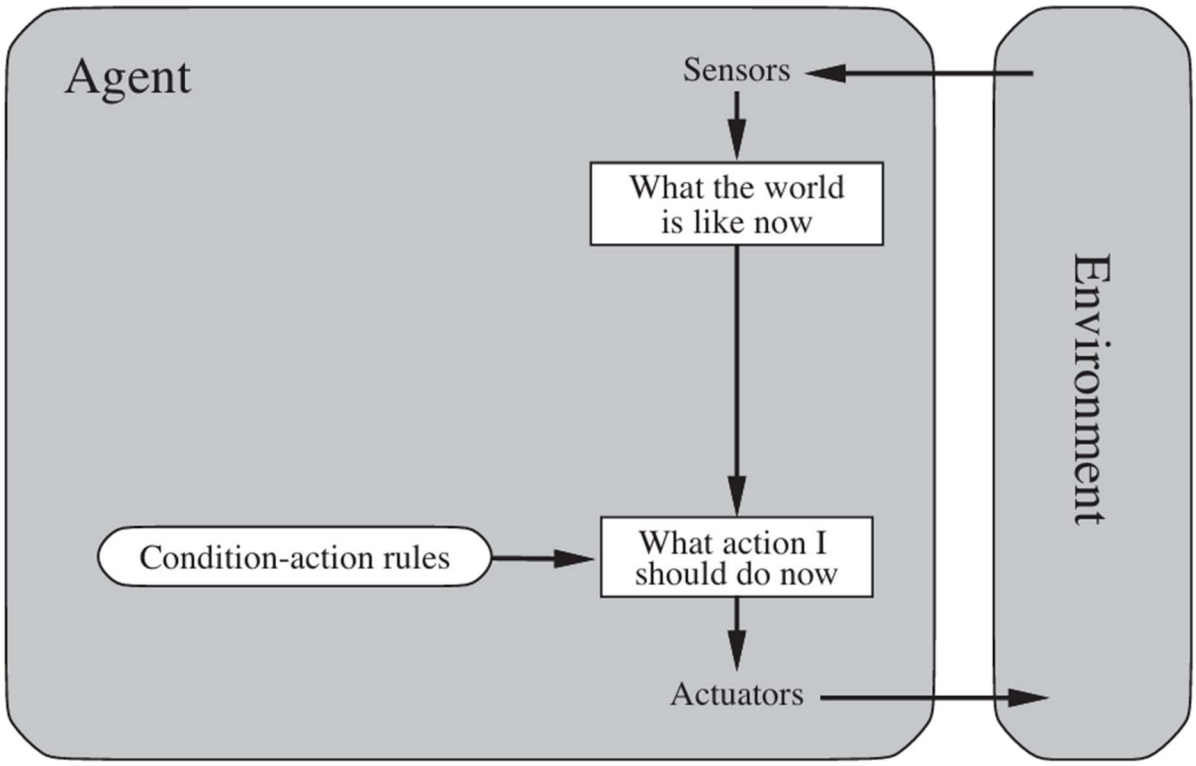
Key Characteristics:

- Has the ability to learn and adapt without being explicitly programmed for every situation.
- Can adjust its behaviour based on past successes or failures.
- Continuously improves through trial-and-error or supervised learning.
- Includes a learning element, a performance element, a critic, and a problem generator.

Example:

A spam email filter that learns to classify new types of spam over time based on user feedback.

Simple Reflex Agent



A Simple Reflex Agent is the most basic type of intelligent agent in Artificial Intelligence. It functions by responding directly to current percepts (inputs from the environment) using predefined condition–action rules—commonly known as if-then rules.

Key Characteristics:

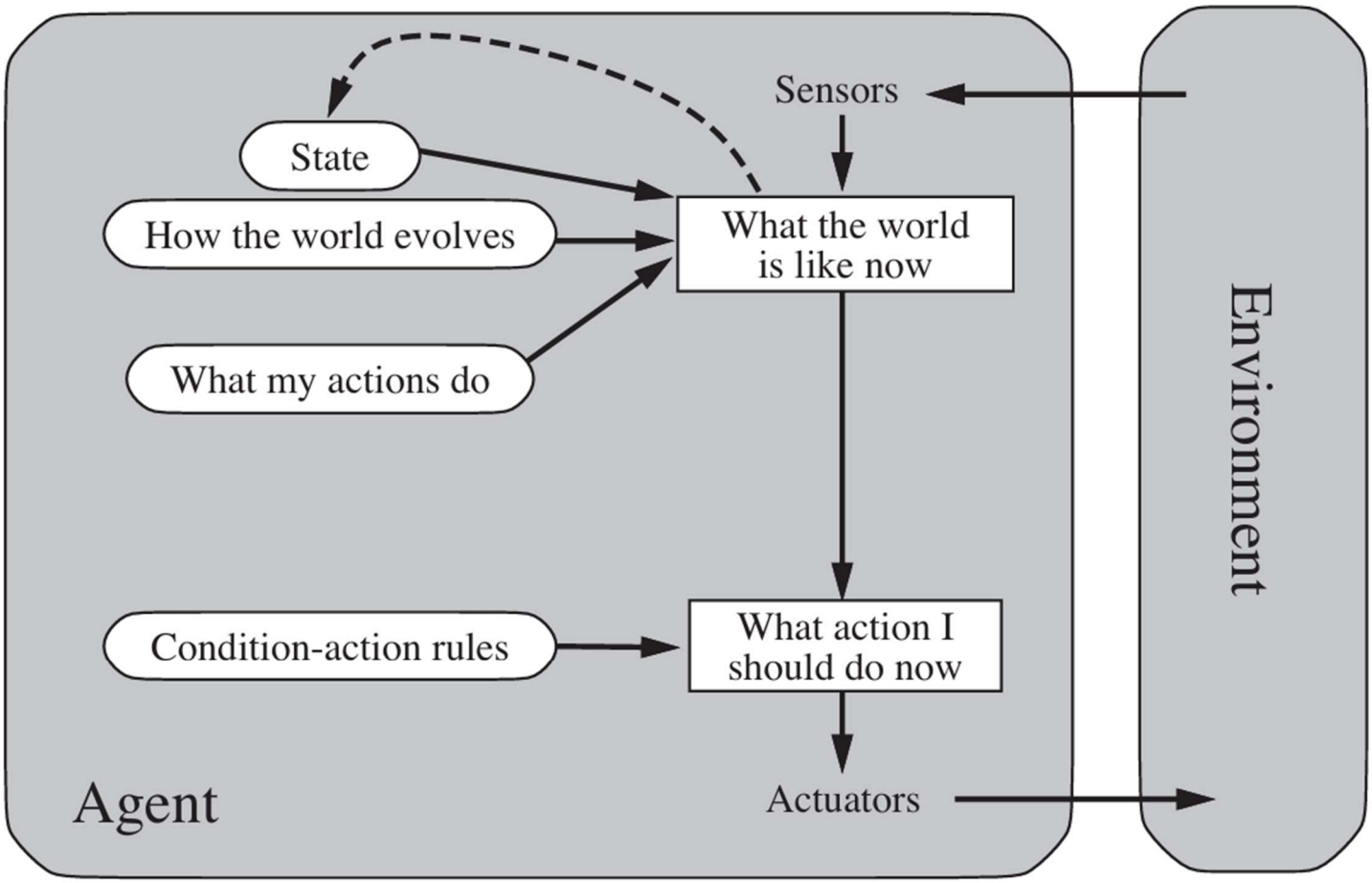
- It does not store past information; decisions are made solely based on the current input.
- Operates using simple condition-action mappings like “if condition then action”.
- No learning or internal model of the world.
- Best suited for fully observable, simple environments.

Example:

A thermostat:

- *If temperature < 20°C → Turn on heater*
- *If* temperature > 25°C → Turn off heater

Model based reflex agent



A Model-Based Reflex Agent is an improvement over the simple reflex agent. It can handle partially observable environments by maintaining some internal state that reflects the unseen aspects of the current world.

Key Characteristics:

- Maintains an internal model of the environment.
- Updates its internal state based on percept history and the agent's actions.
- Uses condition-action rules like simple reflex agents, but with more context.
- Can make better decisions in dynamic or partially observable settings.

Example:

A vacuum cleaner robot that remembers which rooms were already cleaned, even if it can’t see them now.

6. Explain various properties of Task environment with suitable examples.

The task environment defines the problem that an agent is designed to solve. It consists of everything the agent interacts with to perform its job. Various properties help classify and analyse these environments:

1. Fully Observable vs. Partially Observable

- **Fully Observable**: The agent's sensors give access to the entire environment state at any time.
    - **Example**: Chess – all positions of pieces are visible to both players.
- **Partially Observable**: Sensors provide incomplete or noisy data about the environment.
    - **Example**: Autonomous car – cannot always see around corners or in fog.

2. Deterministic vs. Stochastic

- **Deterministic**: The next state of the environment is completely determined by the current state and agent's actions.
    - **Example**: Puzzle games – like the 8-puzzle.
- **Stochastic**: The environment has randomness; the same action may lead to different outcomes.
    - **Example**: Ludo (or any other game involving a dice).

3. Episodic vs. Sequential

- **Episodic**: Agent's experience is divided into independent episodes, and current actions do not affect future ones.
    - **Example**: Image recognition – classifying one image doesn't depend on the previous one.
- **Sequential**: Current decisions can affect future states and decisions.
    - **Example**: Chess – each move affects future outcomes.

4. Static vs. Dynamic

Static: The environment doesn't change while the agent is thinking.

- **Example**: Crossword puzzle – nothing changes unless the agent acts.
- **Dynamic**: The environment changes over time, even if the agent does nothing.
    - **Example**: Autonomous driving – traffic conditions change constantly.

5. Discrete vs. Continuous

- **Discrete**: Environment has a finite number of states, percepts, and actions.
    - **Example**: Board games like tic-tac-toe that have limited moves.

- **Continuous**: States and actions are measured on a continuous scale.
    - **Example**: Robotic arm movement or drone flight.

6. Single Agent vs. Multi-Agent

- **Single Agent**: Only one agent is acting in the environment.
    - **Example**: Vacuum cleaning robot.
- **Multi-Agent**: Multiple agents interact or compete/cooperate.
    - **Example**: Online multiplayer game.

7. What do you mean by state space representation? Explain with example the necessity of it.

State Space Representation is a mathematical model used in AI to describe all the possible states an agent or system can be in, along with the actions that transition the system from one state to another.

It forms the foundation of problem-solving in AI by representing problems as a search through different states to reach a goal.

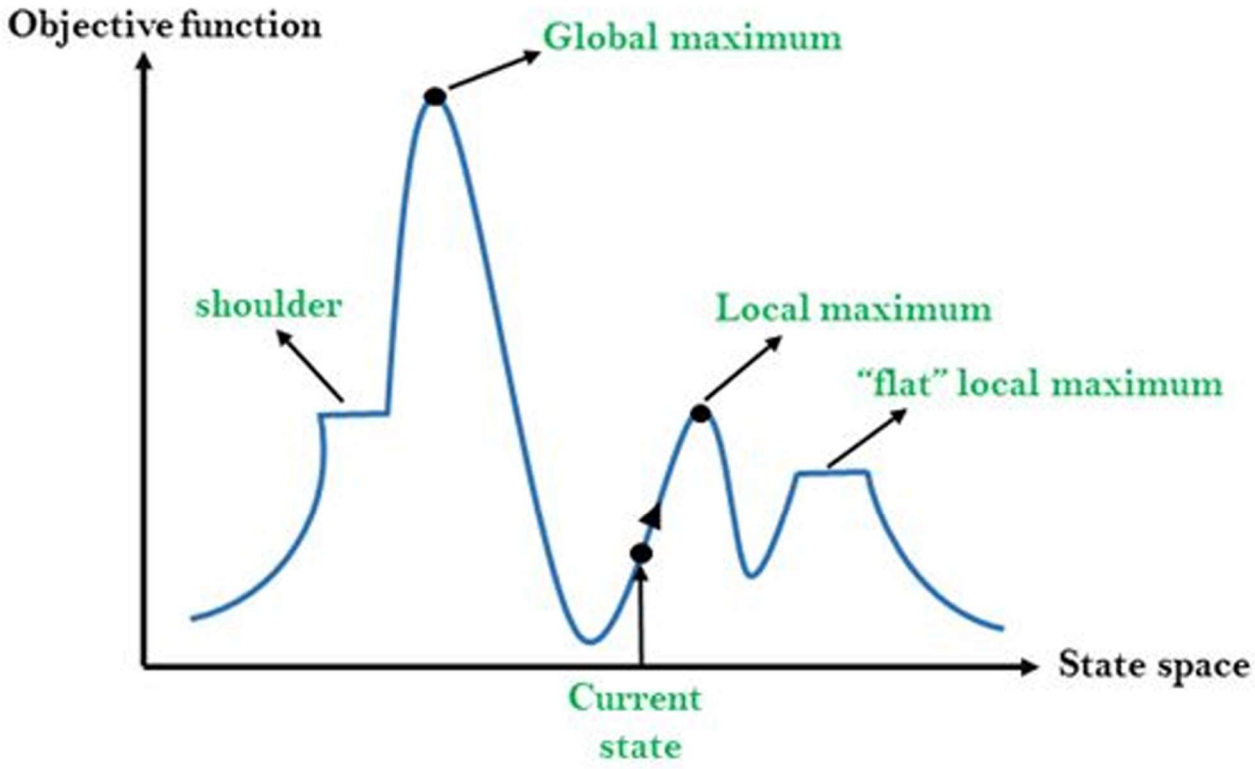
Example: 8-Puzzle Problem

- **Initial State**: A 3×3 board with tiles randomly placed.
- **Goal State**: Tiles arranged in order from 1 to 8 with the blank at the bottom-right.
- **States**: Every legal configuration of tiles on the board.
- **Actions**: Move blank tile up, down, left, or right.
- **Path**: Sequence of moves to reach the goal.
- **Cost**: Each move may have a cost (e.g., 1 per move).

The 8-puzzle problem involves sliding tiles on a 3×3 board to reach a target configuration. Solving this problem manually or with brute force is difficult due to the many possible arrangements (states). Without state space representation, the 8-puzzle cannot be solved efficiently using AI techniques. It transforms the problem into a structured search problem, enabling agents to make intelligent decisions.

3. Problem Solving

8. Explain hill climbing algorithm and problems that occur in hill climbing algorithm along with solutions.



Hill Climbing is a heuristic search algorithm used in Artificial Intelligence for mathematical optimization problems. It's an iterative algorithm that starts with an arbitrary solution and tries to improve it by incrementally changing a single element of the solution.

Different regions in the state space landscape:

1. **Local Maximum:** Local maximum is a state which is better than its neighbour states, but there is also another state which is higher than it.
2. **Global Maximum:** Global maximum is the best possible state of state space landscape. It has the highest value of objective function.
3. **Current state:** It is a state in a landscape diagram where an agent is currently present.
4. **Shoulder:** It is a flat region which has an uphill edge.
5. **Flat local maximum**: It is a flat space in the landscape where all the neighbour states of current states have the same value.

Working of Hill Climbing Algorithm:

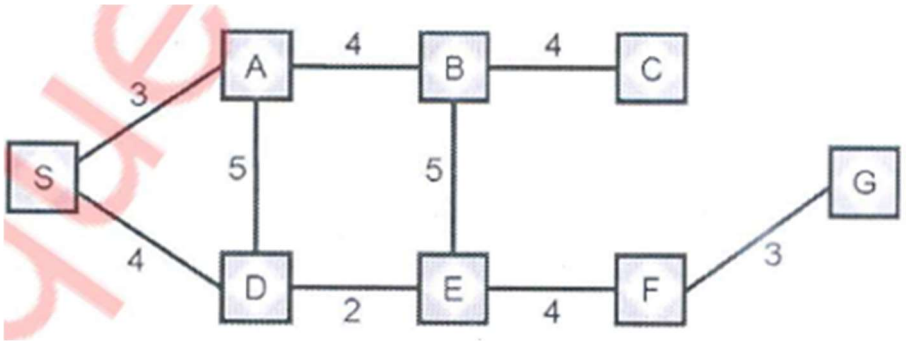
1. Start with an initial solution/state.
2. Evaluate neighbouring states.
3. Move to the neighbour with the highest value (best improvement).
4. Repeat steps 2–3 until:
    - No neighbour is better (local maximum reached)
    - A goal state is found
    - A time or step limit is reached

Problems faced by hill climbing algorithm along with their solutions:

1. **Gets Stuck in Local Optima** – It may find a locally optimal solution instead of the global best solution.
   Solution: Try the algorithm from different starting points until you find a better global solution.
2. **Shoulder Problem** – A flat slope with a gentle increase; hard for the algorithm to detect slow improvement.
   Solution: Use small but multiple moves to detect slight improvements.
3. **Plateau Problem** – If the algorithm reaches a flat region (plateau) in the search space, it may stop making progress.
   Solution: Use sideways moves (limit how many) or introduce small randomness.
4. **Ridge Problem** – If the optimal path requires moving downward temporarily, hill climbing cannot handle it.
   Solution: Use stochastic hill climbing or look ahead multiple steps instead of just one.

9. Apply A* algorithm on a given graph.

PYQ: Heuristic values are h(S)=15, h(A)=14, h(D)=12, h(B)=10, h(E)=10, h(C)=8, h(F)=10, h(G)=0. S is the start node and G is the goal node.



(Asked twice)

Ex. 2.16.1 : Consider the graph given in Fig. P.2.16.1 below. Assume that the initial state is S and the goal state is 7. Find a path from the initial state to the goal state using A* Search. Also report the solution cost. The straight line distance heuristic estimates for the nodes are as follows : h(1) = 14, h(2) = 10. h(3) = 8, h(4) = 12, h(5) = 10, h(6) = 10, h(S) = 15. **MU - Dec. 15, 5 Marks**

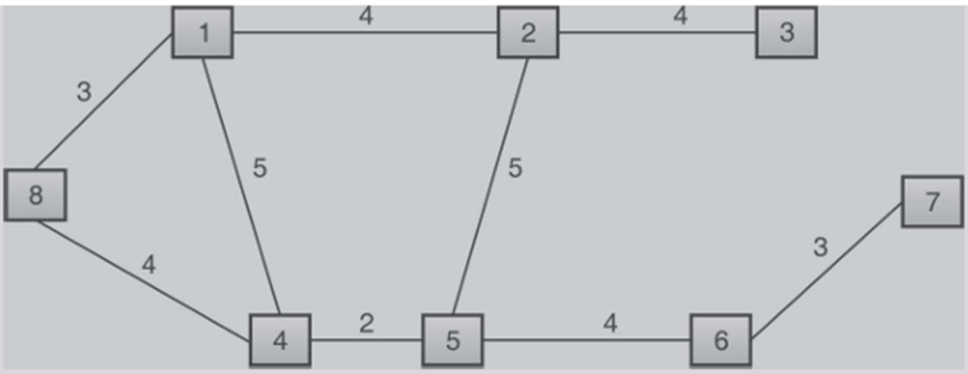
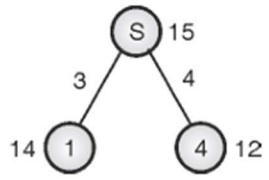
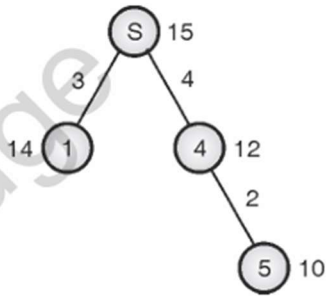


Fig. P. 2.16.1



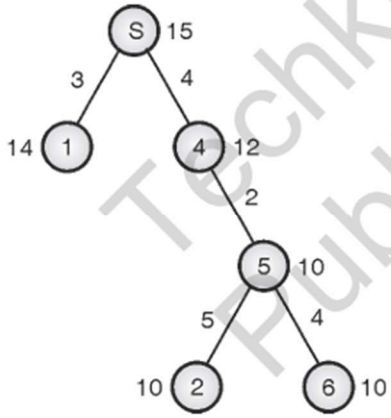
Open : 4(12 + 4), 1(14 + 3)

Closed : S(15)



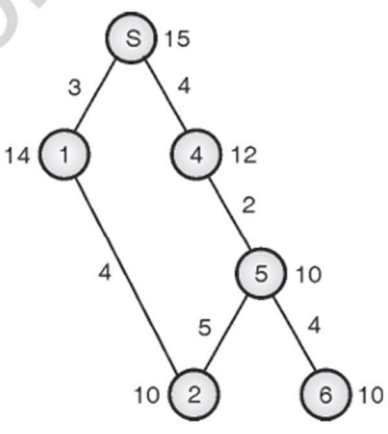
Open : 5(10 + 6), 1(14 + 3)

Closed : S(15), 4 (12 + 4)



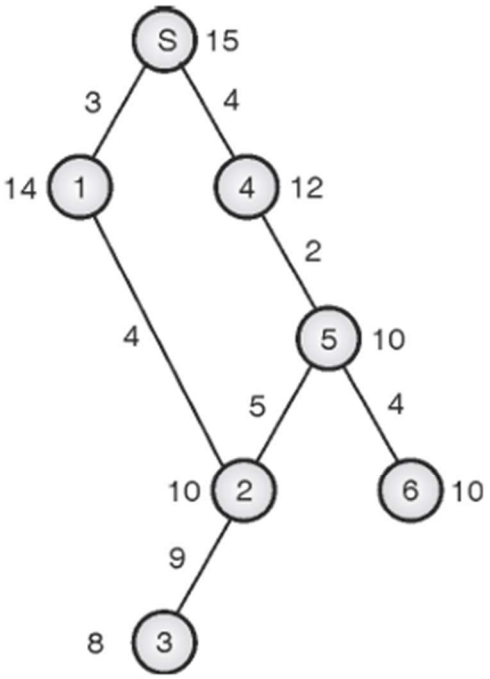
Open : 1(14 + 3) 6(10 + 10) 2(10 + 11)

Closed : S(15) 4(12 + 4) 5(10 + 6)



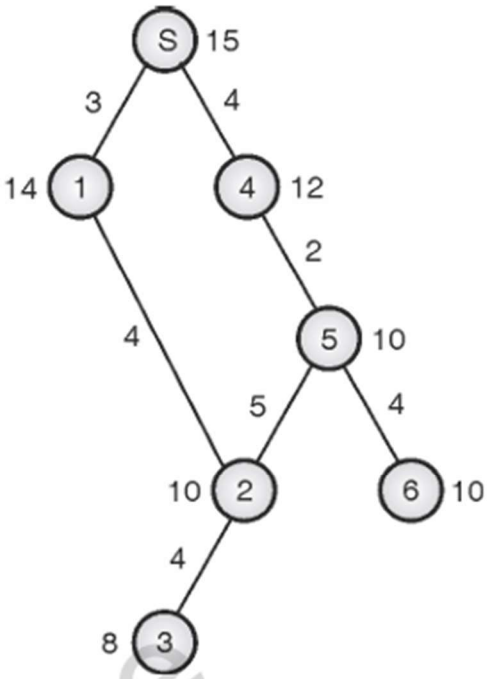
Open : 2(10 + 7) 6(10+10)

Closed : S(5) 4(12 + 4) 5(10 + 6) 1(14 +3)



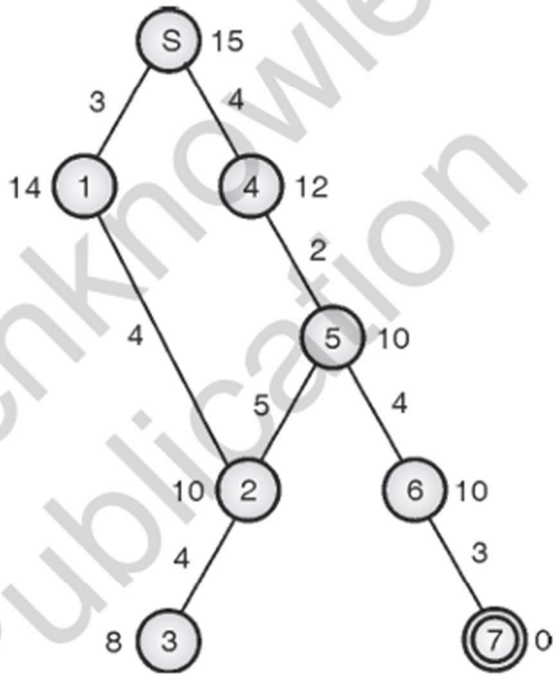
Open : 3(8 +11), 6(10 + 10)

Closed : S(15),4(12+4),5(10+6),
1(14+ 3),2(10+7)



Open : 6(10 + 10)

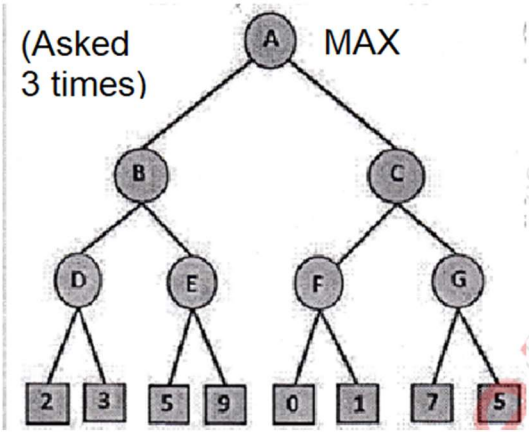
Closed : s(15), 4(12 + 4), 5(10 + 6),
1(14 + 3) 2(10 + 7) 3(8 + 11)

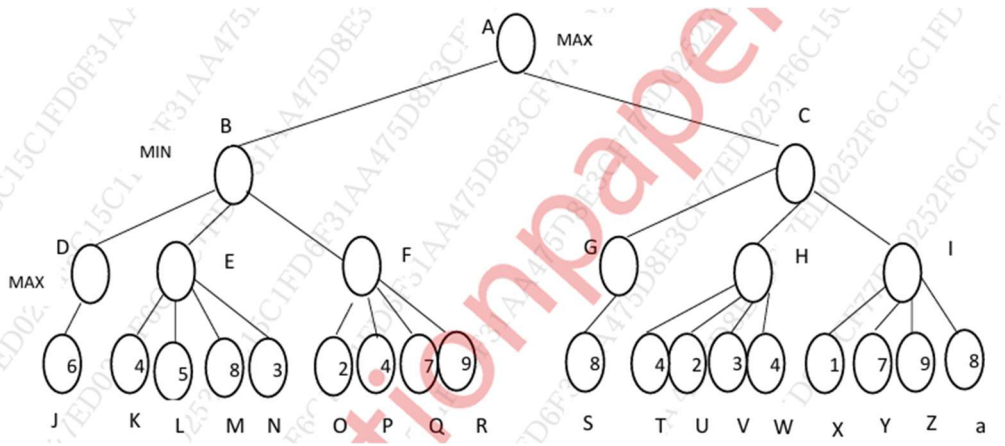


Open : 7(0 + 13)

Closed : S(15), 4(12 + 4), 5(10 + 6), 1(14 + 3), 2(10 + 7), 5(8 + 4), 6(10 + 10)

10. Apply alpha beta pruning on given graph, PYQ





(Explanation for understanding, not required)

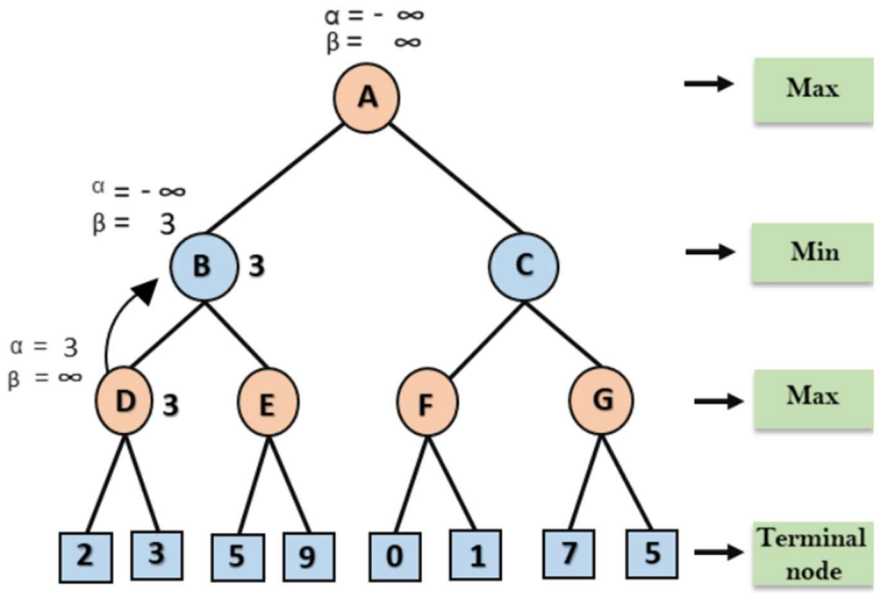
Step 1: Max starts at node A with α = −∞ and β = +∞. These values are passed to node B and then to its child D.

Step 2: At node D (Max's turn), α is updated by comparing 2 and 3 → max(2, 3) = 3.

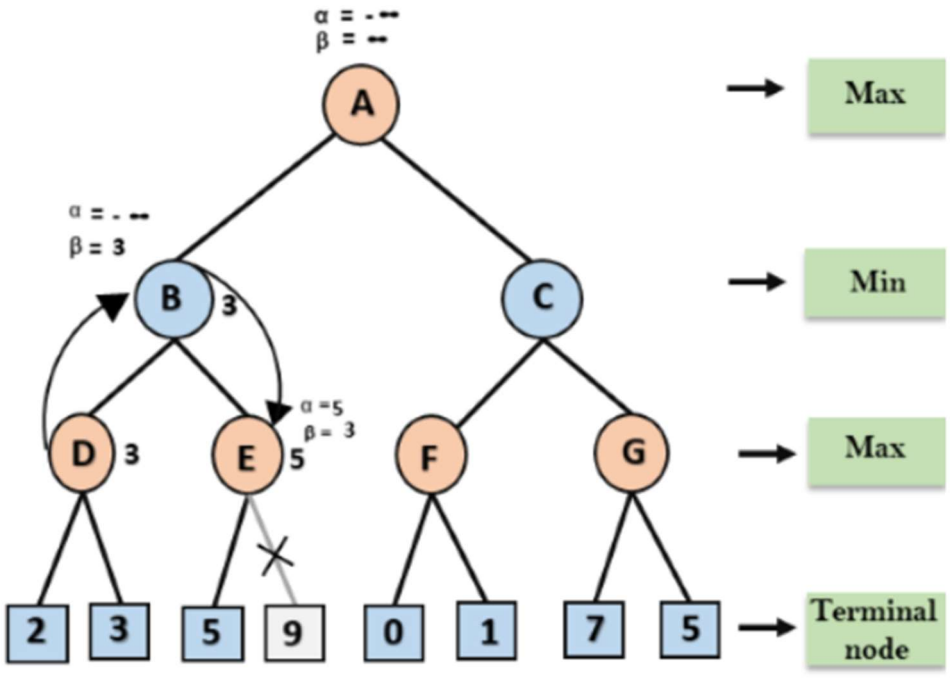
So, α = 3 at D.

Step 3: Back at node B (Min's turn), β is updated → min(+∞, 3) = 3.

So, B now has α = −∞ and β = 3. Next, node E is explored with the same values passed.

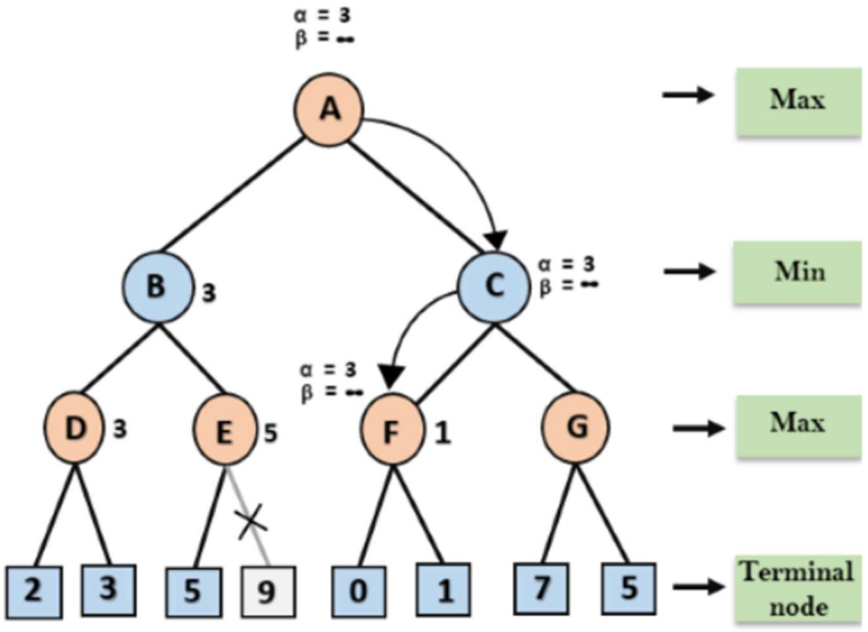


Step 4: At node E (Max's turn), α = max(−∞, 5) = 5. Since α > β (5 > 3), pruning occurs — the right child of E is not explored. Node E value = 5.



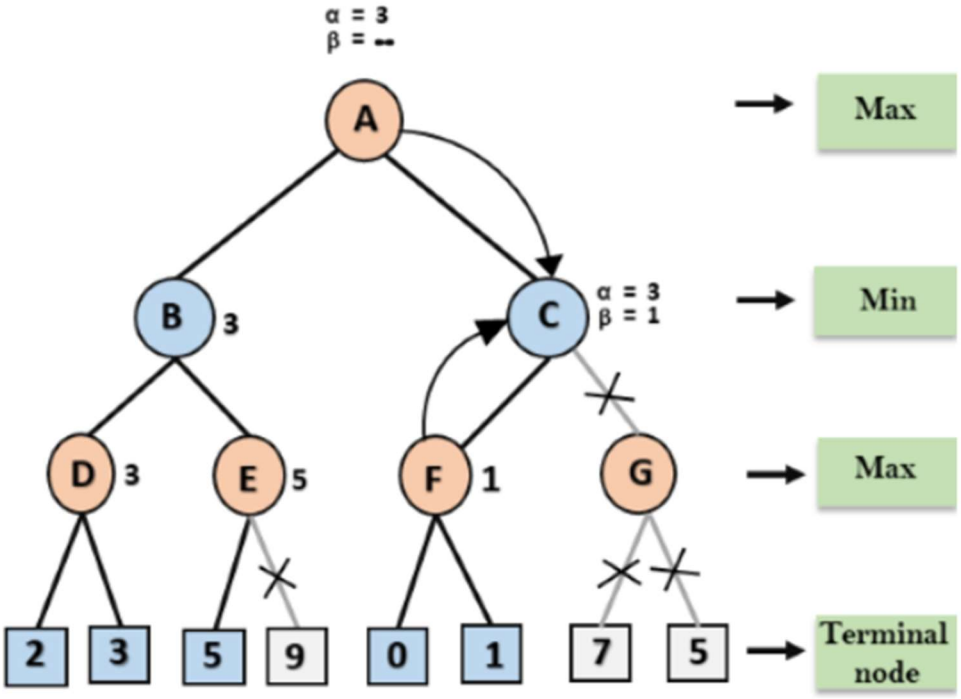
Step 5: Backtrack to A. At A, α = max(−∞, 3) = 3. β = +∞. These values are passed to node C.

Step 6: At node F (Max's turn), α = max(3, 0) = 3, then compared with 1 → max(3, 1) = 3. Node F value = 1.

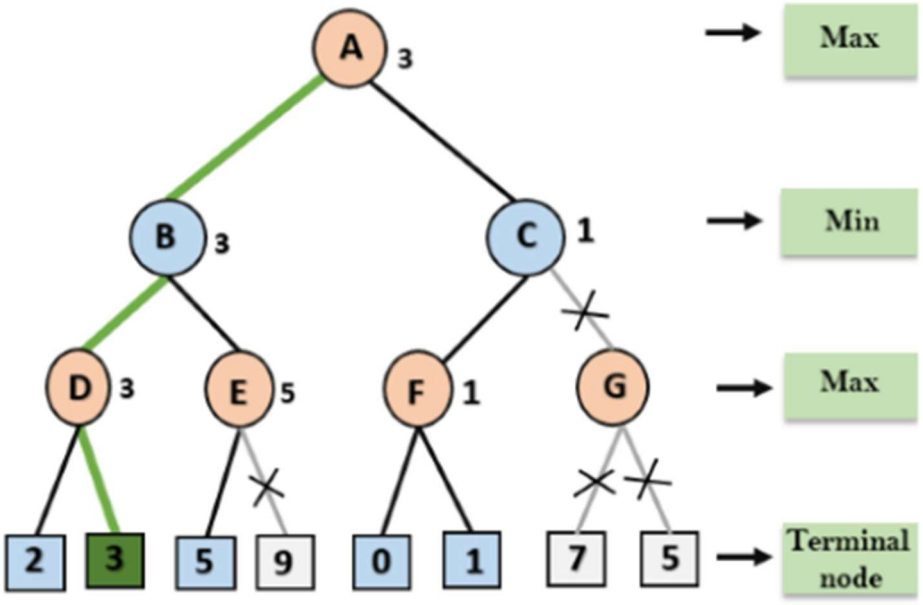


Step 7: F returns 1 to C. β = min(+∞, 1) = 1.

Since α ≥ β(3≥1), the next child (G) is pruned.



Step 8: C returns 1 to A. Final value at A = max(3, 1) = 3. This is the optimal value for Max. The final tree shows which nodes were computed and which were pruned.



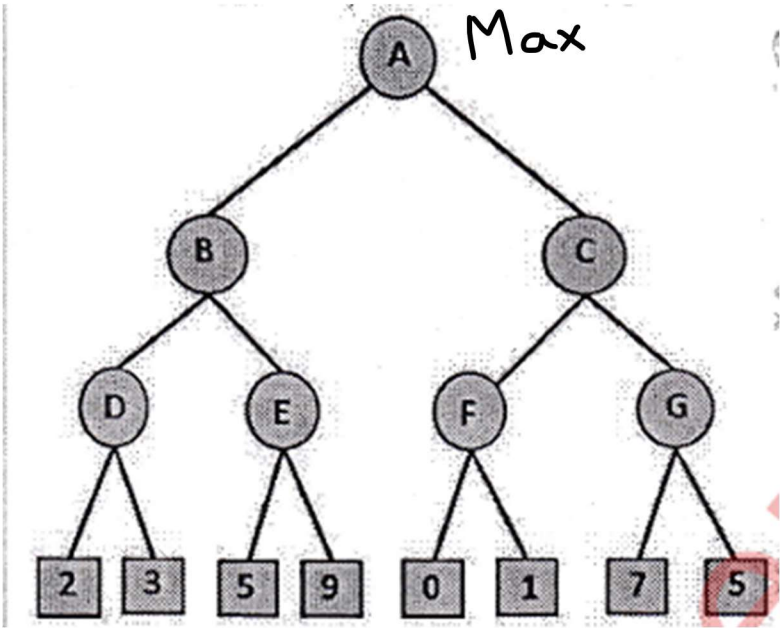
11. Explain alpha beta search in detail with example.

Alpha-Beta Pruning is an optimization technique for the Minimax algorithm, used in decision-making and game theory. It reduces the number of nodes evaluated in the search tree by pruning branches that won't affect the final decision.

Concepts:

- **Min max Algorithm**: Used to determine the best move for a player assuming the opponent also plays optimally.
- **Alpha (α)**: Best (highest) value the maximizer currently can guarantee at that level or above.
- **Beta (β)**: Best (lowest) value the minimizer currently can guarantee at that level or above.

Example: (Same as before)



Step-by-Step Alpha-Beta Pruning

Step 1: A (MAX)

- α = −∞, β = +∞
- Go to B

Step 2: B (MIN)

- α = −∞, β = +∞
- Go to D

Step 3: D (MAX)

- Children: 2 and 3 → max(2, 3) = 3
- Return 3 to B → B's β becomes 3

Step 4: E (MAX)

- First child = 5 → α = 5
- B's β = 3 → now **α (5) ≥ β (3)** → prune 9
- Return 5 to B → B chooses min(3, 5) = **3**

Step 5: A (MAX)

- Receives 3 from B → α = 3
- Go to C

Step 6: C (MIN)

- α = 3, β = +∞
- Go to F

Step 7: F (MAX)

- Children: 0 and 1 → max(0, 1) = 1
- Return 1 to C → β = 1
- Now at A: α = 3, β = 1 → **α ≥ β**, prune G

Result:

- Final value at A: **max(3, 1) = 3**
- Nodes pruned: 9 (under E), and both 7 & 5 (under G)

Advantages:

1. **Faster Decision-Making:**
   It reduces the number of nodes evaluated, making the algorithm significantly faster.
2. **Same Optimal Result as Minimax:**
   It gives the same result as Min max but with fewer calculations.

Disadvantages:

1. **Depends on Move Order:**
   Pruning is most effective only if the best moves are considered first. Poor move ordering reduces its benefits.
2. **Harder to Implement Than Minmax:**
   Requires careful management of alpha and beta values, making it slightly more complex.

12. Differentiate between Informed search and Uninformed search algorithms. #

Parameter	Uninformed Search	Informed Search
Definition	Searches without any domain-specific knowledge	Uses domain-specific knowledge (heuristics) to guide the search
Also Known As	Blind search	Heuristic search
Use of Heuristics	Does not use heuristics	Uses heuristics to estimate cost to goal
Efficiency	Less efficient, explores more nodes	More efficient, explores fewer nodes
Time and Space Complexity	Generally higher	Lower (depending on the heuristic's quality)
Goal Direction	Not goal-directed	Goal-directed
Accuracy in Reaching Goal	May reach goal but not always in optimal way	Often finds optimal or near-optimal path
Examples	Breadth-First Search (BFS), Depth-First Search (DFS), Uniform Cost Search	A* Search, Greedy Best-First Search

13. Explain the Depth Limited search and Depth first iterative deepening search.

1. Depth-Limited Search (DLS)

Definition:

A depth-first search algorithm with a fixed limit on how deep the search tree can go. It avoids exploring nodes beyond a specified depth l.

Termination Condition:

- If the goal is found within the depth limit → **Success**
- If depth limit is reached without finding the goal → **Cutoff/Failure**

Advantages:

- Prevents infinite loops in deep or infinite search trees.
- Uses less memory

Disadvantages:

- Can be terminated without finding solution
- Not optimal.

Example:
Searching in a tree up to depth l = 3, it will ignore any paths deeper than 3 levels.

2. Depth-First Iterative Deepening Search (DFID)

Definition:

A search strategy that repeatedly applies depth-limited search with increasing depth limits (0, 1, 2, ...) until the goal is found.

Termination Condition:

- Goal is found at some depth → **Success**
- All nodes are explored without finding the goal → **Failure**

Advantages:

- Complete and optimal
- Memory-efficient
- Doesn't require prior knowledge of depth.

Disadvantages:

- Repeats exploration of nodes from previous levels (redundant work).
- Slower than BFS for shallow goals due to repetition.

Example:
Searches the tree first to depth 0, then 1, then 2, and so on, until it finds the goal.

14. Explain Simulated annealing with suitable example.

Simulated Annealing is a probabilistic optimization algorithm inspired by the annealing process in metallurgy. It is used to find an approximate global optimum of a problem by occasionally allowing worse moves to escape local optima.

Key Concept:

- **Current state**: The current solution.
- **Neighbouring state**: A slightly changed version of the current solution.
- **Temperature (T)**: Controls the probability of accepting worse solutions.
- As T decreases, the algorithm becomes less likely to accept worse moves.

Steps:

1. Start with an initial solution.
2. Set initial temperature T.
3. Repeat until stopping condition is met:
    - Generate a neighbouring solution.
    - If it's better, accept it.
    - If it's worse, accept it with probability: $P=e^{-\Delta E/T}$
      where ΔE = change in cost or fitness.
    - Gradually reduce T.

Example – Traveling Salesman Problem (TSP)

Problem: A salesman must visit several cities once and return to the starting point with the shortest total distance.

- **Initial solution**: A random path through all cities.
- **Neighbour**: A new path by swapping two cities.
- **Cost**: Total distance of the path.

Annealing Schedule:

- Start with a high temperature, allowing the algorithm to accept worse solutions (i.e., longer distances) to escape local minima.
- Gradually cool down the temperature over time, making the algorithm less likely to accept worse solutions.
- Eventually, it finds better paths, converging toward the global optimum.

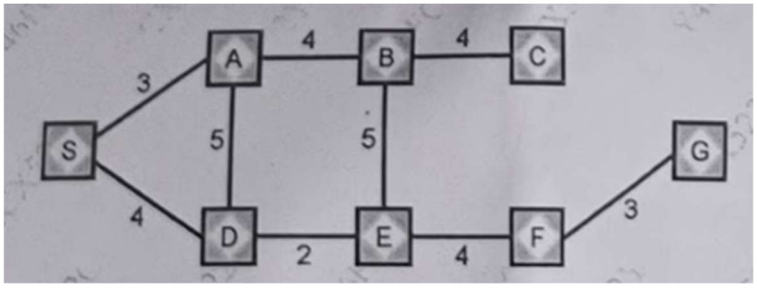
Advantages:

- Can escape local optima.
- Simple to implement.

Disadvantages:

- Requires good cooling schedule.
- Slower than greedy algorithms.

15. Apply greedy best-first search. At each iteration, each node is expanded using evaluation function f(n) = h(n). h(S)=10, h(A)=10, h(D)=8, h(B)=6, h(E)=6.5, h(C)=4, h(F)=3, h(G)=0. S is the start state and G is goal state.



Greedy Best-First Search uses the evaluation function:

f(n)=h(n)

Given:

- Start node = **S**
- Goal node = **G**
- Heuristic values:
  h(S)=10, h(A)=10, h(D)=8, h(B)=6, h(E)=6.5, h(C)=4, h(F)=3, h(G)=0
- Numbers on the edges of the graph (like 3, 4, 5, etc.) represent the step cost or actual path cost g(n), however, in Greedy Best-First Search, these edge costs are ignored.

Step-by-step Expansion:

Step 1: Start at **S**, add its children A and D to the open list.

- h(A) = 10
- h(D) = 8
  Choose **D** (lowest h)

Step 2: From **D**, expand children: **E**

- h(E) = 6.5
  Open list: A(h=10), E(h=6.5)
  Choose **E**

Step 3: From **E**, expand children: **B** and **F**

- h(B) = 6
- h(F) = 3
  Open list: A(10), B(6), F(3)
  Choose **F**

Step 4: From **F**, expand child: **G**

- h(G) = 0
  Open list: A(10), B(6), G(0)
  Choose **G**

Goal Reached

Path Traversed:

S → D → E → F → G

Heuristic Order Used:

10 → 8 → 6.5 → 3 → 0

16. What is Game playing algorithm? Draw a game tree for Tic-Tac-Toe problem.

A Game Playing Algorithm is an artificial intelligence technique used to make optimal decisions in competitive games like Tic-Tac-Toe, Chess, or Checkers. These algorithms simulate all possible moves and outcomes using a game tree, helping the agent select the best move based on strategic evaluation.

Key Concepts:

- **Game Tree**: A tree structure showing all possible sequences of moves.
- **Minimax Algorithm**: Selects moves to maximize a player's gain and minimize the opponent's gain.
- **Alpha-Beta Pruning**: Improves efficiency by eliminating branches that won't affect the final decision.
- **Utility Function**: Scores game outcomes (e.g., win = +1, loss = -1, draw = 0).

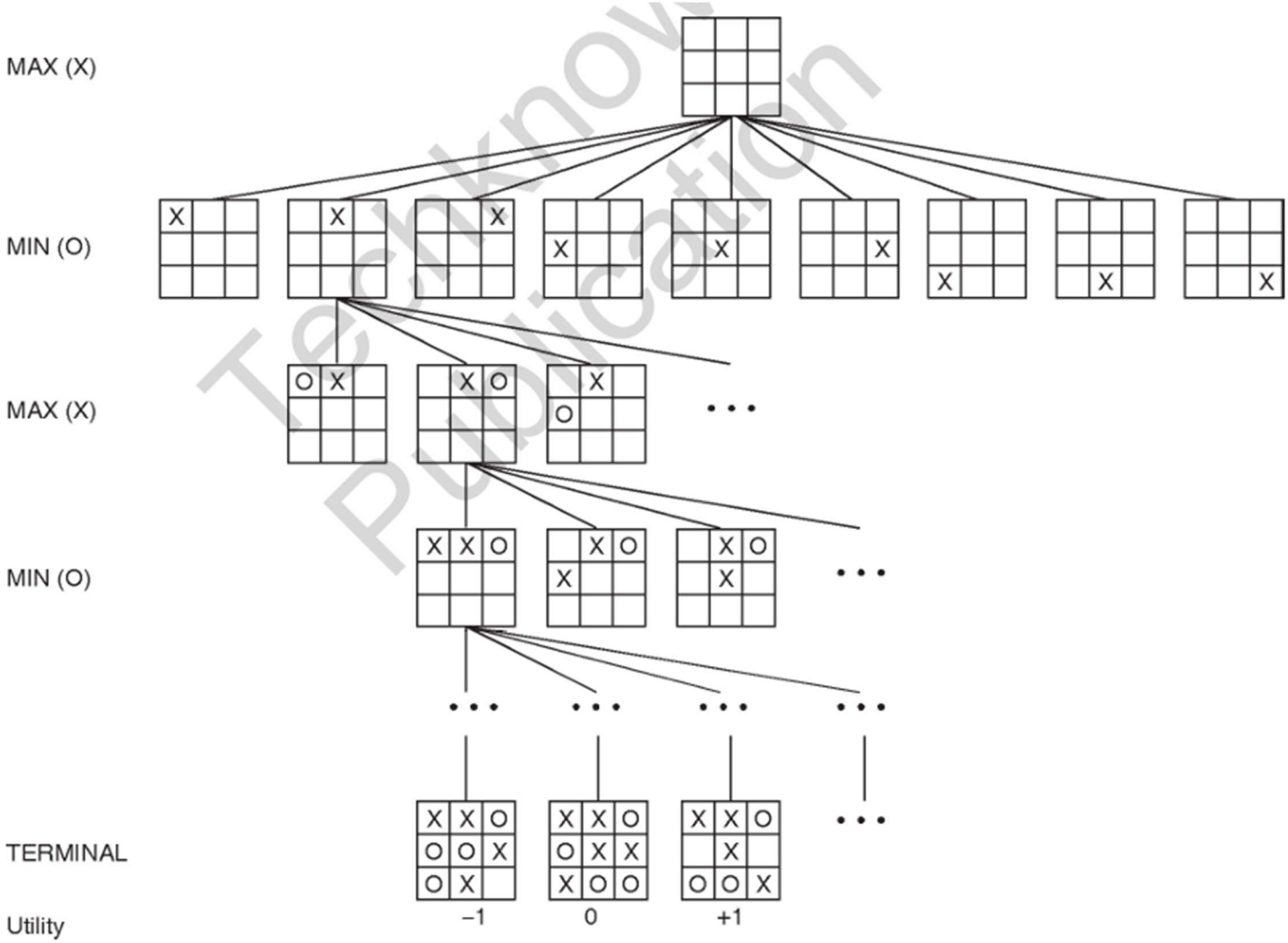


Fig. 2.20.3 : Tic-tac-toe game tree

17. What do you understand by Min Max Search? Explain with example.

The Minimax algorithm builds a game tree starting from the current state of the game. The tree explores all possible moves until the game ends (either with a win, loss, or draw). Each node represents a game state, and the algorithm recursively evaluates the utility values (score) of these terminal states.

Key Points:

- **Maximizing Player (Max)**: The player whose turn it is and aims to maximize their score (e.g., Player X).
- **Minimizing Player (Min)**: The opponent, who tries to minimize the maximizing player's score (e.g., Player O).

The algorithm works by backtracking from the terminal states of the game tree, assigning values (scores) to nodes based on the utility:

- **Max Player**: Chooses the move with the highest score.
- **Min Player**: Chooses the move with the lowest score.

Example:

1. B is a Min node → chooses min(3, 5) = 3
2. C is a Min node → chooses min(2, 9) = 2
3. A is a Max node → chooses max(3, 2) = 3

Result: Max chooses the path through B, expecting to get a value of 3.

Applications:

- Chess, Tic-tac-toe, Checkers

18. What do you understand by informed and uninformed search methods? Explain in detail with example.

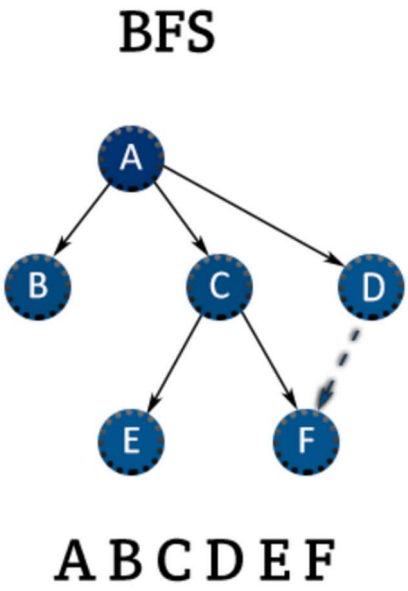
1. Uninformed Search (Blind Search)

Definition:

Uninformed search methods do not have any domain-specific knowledge about the goal's location. They only use the problem definition (like initial state, goal test, and successor function).

Examples:

- Breadth-First Search (BFS)
- Depth-First Search (DFS)
- Uniform Cost Search
- Depth-Limited Search



Example – Breadth-First Search (BFS):

BFS explores nodes level-by-level.

Level 0: A

Level 1: A→B→C→D

Level 2: A→B→C→D→E→F

Final path: A→B→C→D→E→F

Advantages:

- Simple and easy to implement
- Guaranteed to find a solution (if one exists)

Disadvantages:

- Inefficient (explores a lot of unnecessary paths)
- Time and space-consuming

2. Informed Search (Heuristic Search)

Definition:

Informed search uses additional knowledge or heuristics to make more intelligent decisions about which node to explore next.

A heuristic function (h(n)) estimates the cost from the current node to the goal.

Examples:

- Best-First Search
- A* Search

- Greedy Search

Example – A* Search:

Uses both actual cost (g(n)) and heuristic (h(n)):

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

If you're using a city map and want to reach City Z, A* would choose roads that are likely to get you there faster, not just the first available.

Advantages:

- Faster and more efficient
- Uses domain knowledge to guide the search

Disadvantages:

- Requires a good heuristic
- More complex to implement

19. Explain the concept of genetic programming.

Genetic Programming is an evolutionary algorithm-based technique in AI that automatically generates computer programs to solve a given problem. It is inspired by the process of natural selection in biological evolution.

Components of genetic programming:

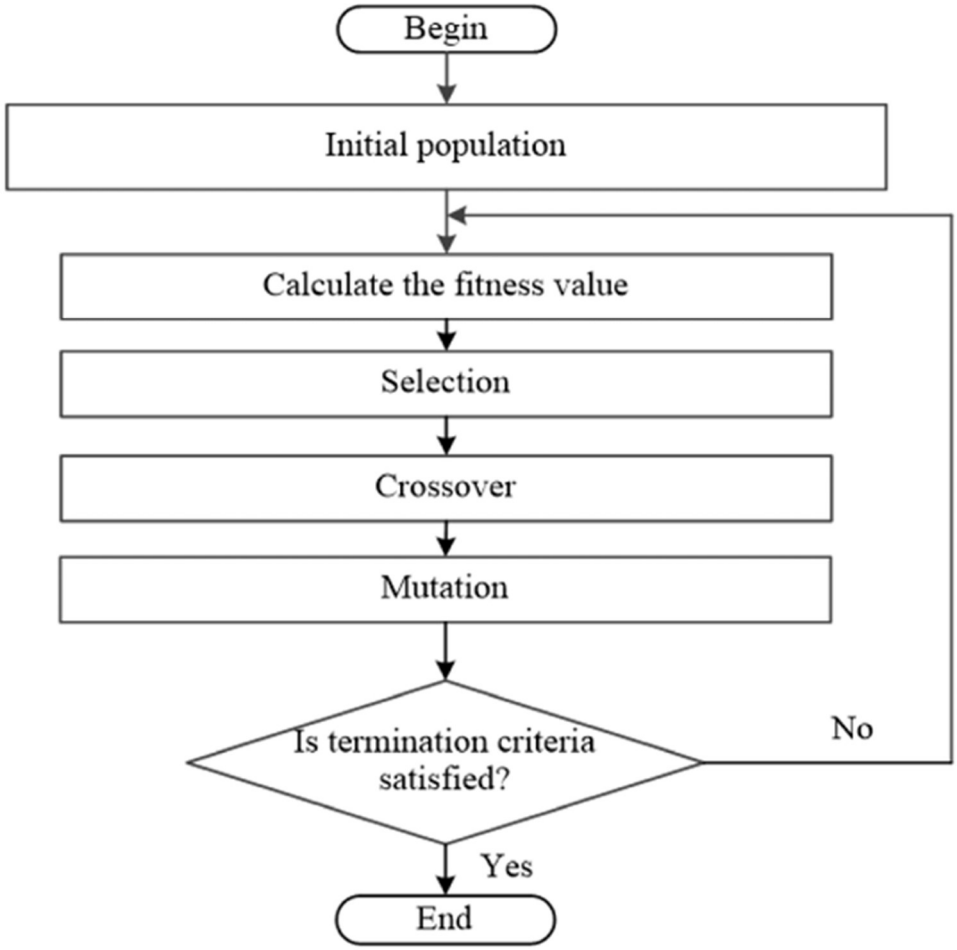
Chromosome – A chromosome represents a possible solution to the problem, encoded as a string of genes (binary, numeric, or symbolic). It undergoes selection, crossover, and mutation to evolve better solutions**.**

Fitness Function – The fitness function evaluates how good a chromosome is in solving the given problem.

Selection – It chooses the best chromosomes based on their fitness to pass genes to the next generation.

Crossover – It combines genetic material from two or more parent chromosomes to create new offspring.

Mutation – It randomly alters genes in a chromosome to introduce diversity.



Working of Genetic Algorithm:

1. **Initialization** – Generate an initial population of chromosomes.
2. **Evaluation** – Compute the fitness of each chromosome using the fitness function.
3. **Selection** – Choose the best individuals for reproduction.

4. **Crossover** – Combine selected parents to create new offspring.
5. **Mutation** – Introduce small random changes to maintain diversity.
6. **Replacement** – Form a new generation by replacing old individuals with new ones.
7. **Termination** – Repeat steps 2-6 until an optimal solution is found or conditions are met.

4. Knowledge and Reasoning

20. Explain forward-chaining and backward-chaining algorithm in detail with suitable example.

1. Forward-Chaining Algorithm

Definition:

Forward-chaining is a data-driven reasoning approach. It starts from known facts and uses inference rules to derive new facts, continuing until it reaches the goal or no new facts can be inferred.

Working:

- Start with a set of facts.
- Apply rules whose conditions match the current facts.
- Add the conclusions of those rules to the fact base.
- Repeat the process until the goal is reached or no more rules apply.

Example of Forward Chaining

Scenario: Identifying an animal

- **Rules:**
    1. If an animal has feathers, it is a bird.
    2. If a bird cannot fly, it is a penguin.
- **Process:**
    - Given: "The animal has feathers."
    - The system deduces: "It is a bird."
    - Next, if "The bird cannot fly," it concludes: "It is a penguin."

2. Backward-Chaining Algorithm

Definition:

Backward-chaining is a goal-driven reasoning approach. It starts from the goal and works backwards, trying to find facts or rules that can support that goal.

Working:

- Start with a goal (what you want to prove).
- Look for rules that could conclude this goal.
- Check if the conditions of those rules are satisfied.
- If not, treat those conditions as sub-goals and repeat the process.

Backward Chaining Example

Goal: Determine if a person is eligible to vote.

- If a person is 18 or older and has valid ID, then they are eligible to vote.

Process: Start with "Is the person eligible to vote?"

Check if they are 18 or older

Verify if they have valid ID

If both conditions hold, conclude they can vote.

21. Write a detailed note on Wumpus world environment.

- The Wumpus World is a grid-based environment used in AI to demonstrate logical reasoning & decision making.
- The world is a typically a 4x4 grid of rooms.
- Each room may contain a pit, the Wumpus (a monster) or gold.
- The agent starts in the bottom-left corner [1,1] & can move one cell at a time.

Goal of the agent :

- The agent's objective is to find the gold & return safely to the starting point
- It must avoid falling into pits or being killed by the Wumpus.

PEAS Properties :

1. Performance Measure:

- +100 points for grabbing the gold & coming back to the starting position
- -1 per action
- -200 if the agent (player) is killed
- -10 if the arrow is used

2. Environment:

- Empty Rooms
- Room with Wumpus
- Stenchy Rooms (rooms neighbouring to Wumpus)
- Breezy rooms (rooms neighbouring to pits)
- Room with gold
- Arrow to shoot the Wumpus

3. Sensors:

- Camera to get the view
- Odour sensor to smell the stench
- Audio sensor to listen to the scream

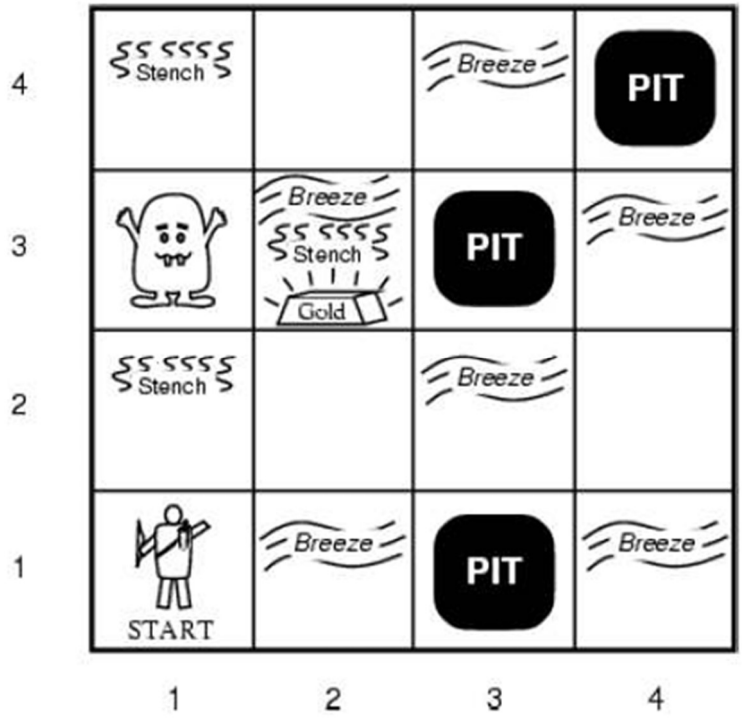
4. Effectors:

- Motor to move left, right
- Robotic arm to grab the gold
- Robotic mechanism to shoot the arrow

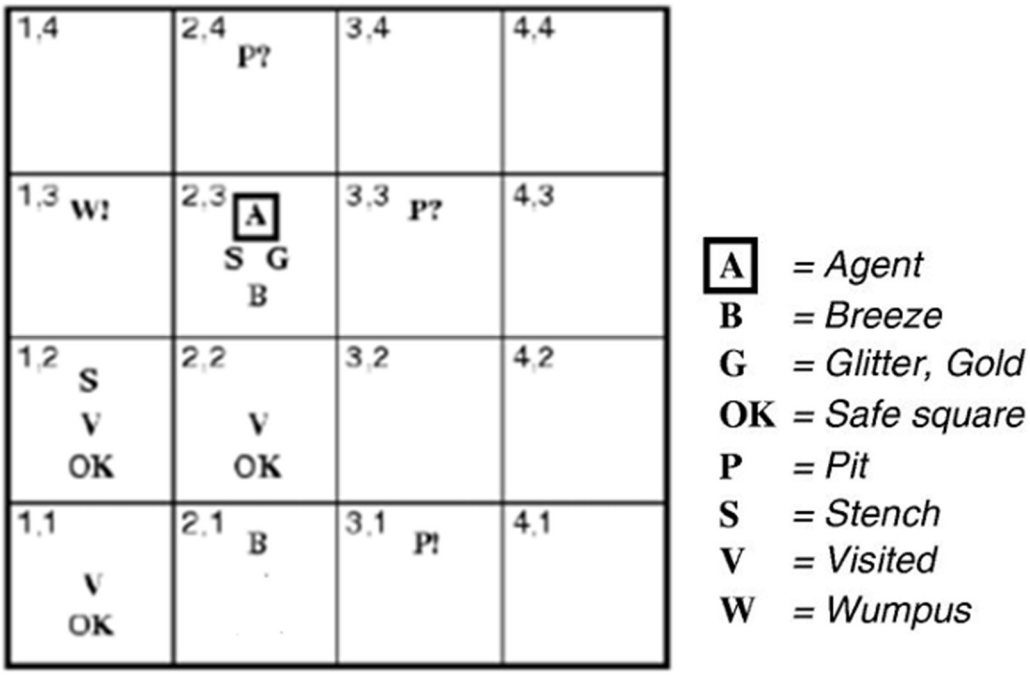
Agent has following characteristics:

- **Agents-** Single agent
- **Observability-** Partially observable (only percepts from adjacent cells)
- **Determinism**- Deterministic (actions have known outcomes)
- **Episodic/Sequential-**Sequential (decisions affect future)
- **Static/Dynamic**- Static (nothing changes while the agent thinks)
- **Discrete/Continuous-** Discrete

Example: Sample WUMPUS world



Agent has the following percepts: Percept (Breeze, Stench, Glitter, Bump, Scream).



The agent follows the path below, experiencing the listed percepts in each room.

Agent's path: (1,1) → (1,2) → (2,2) → (2,3)

The agent avoids rooms with pits and Wumpus by using percepts like Breeze and Stench to infer danger. It moves only into safe or uncertain rooms unless forced.

Percept Sequence:
Percept (1,1)= (None, None, None, None, None)
Percept (1,2)= (None, **Stench**, None, None, None)
Percept (2,2)= (None, None, None, None, None)
Percept (2,3)= (**Breeze, Stench**, **Glitter**, None, None)

The agent reaches the room (2,3), detects Glitter (indicating gold), picks it up, and retraces the path back to (1,1).

22. Solve given problem using Resolution:

i. Consider the following statements: (asked twice)

a) All people who are graduating are happy

b) All happy people smile

c) Someone is graduating

Represent above statements in FOL, Convert each to CNF, Prove that "someone is smiling" using resolution. Draw the resolution tree.

ii. What actions would you take to prove "Some who are intelligent can't read" using propositional logic:

a) Whoever can read is literate

b) Dolphins are not literate

c) Some dolphins are intelligent

iii. Consider the following facts:

a) Steve only likes easy courses.

b) Science courses are hard.

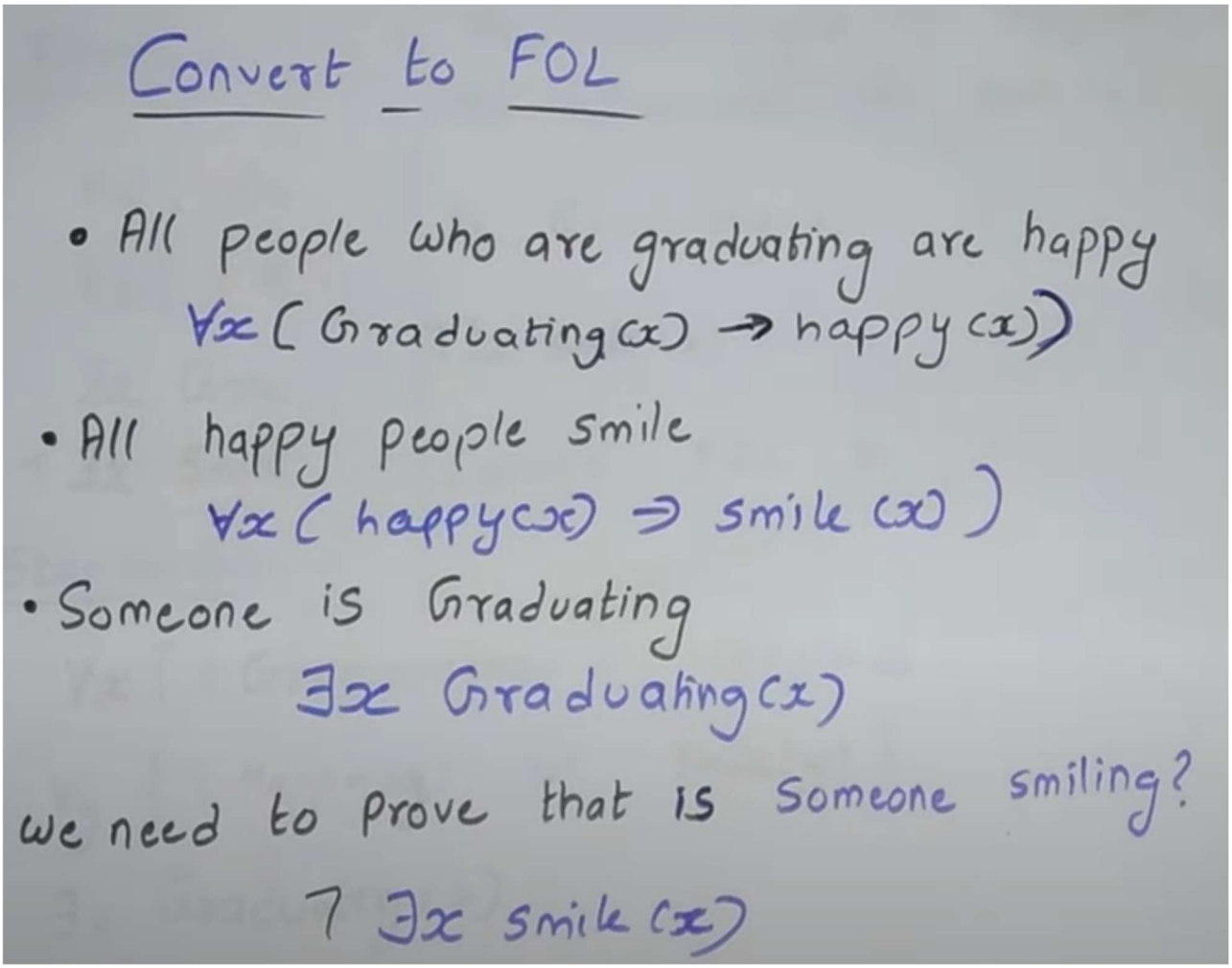
c) All the courses in the basket _ weaving department are easy.

d) BK301 is a basket _ weaving course.

Find by resolution that "What course would Steve like?"

i. https://www.youtube.com/watch?v=IWIP2NuwhBQ

Step 1: Convert to FOL



Step 2: Convert FOL to CNF

Eliminate Implication $\alpha \rightarrow \beta \equiv \neg\alpha \vee \beta$

$\forall x [\neg \text{Graduating}(x) \vee \text{Happy}(x)]$

$\forall x [\neg \text{Happy}(x) \vee \text{Smile}(x)]$

$\exists x \ \text{Graduating}(x)$

$\neg \exists x \ \text{Smile}(x)$

Standardize variables APart

$\forall x [\neg \text{Graduating}(x) \vee \text{Happy}(x)]$

$\forall y [\neg \text{Happy}(y) \vee \text{Smile}(y)]$

$\exists z \ \text{Graduating}(z)$

$\neg \exists w \ \text{smile}(w)$

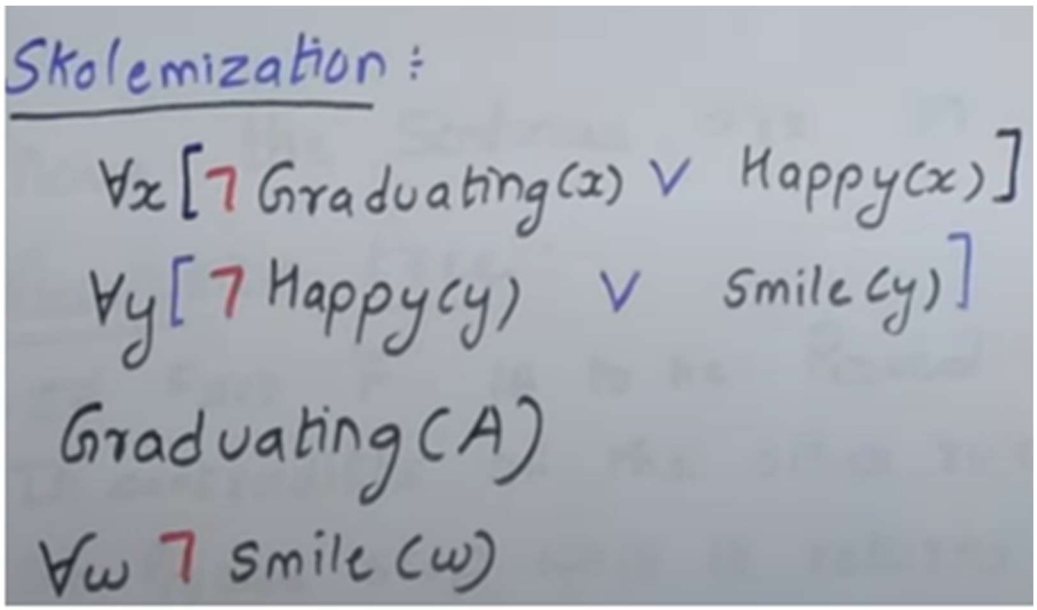
Move Negation Inwards

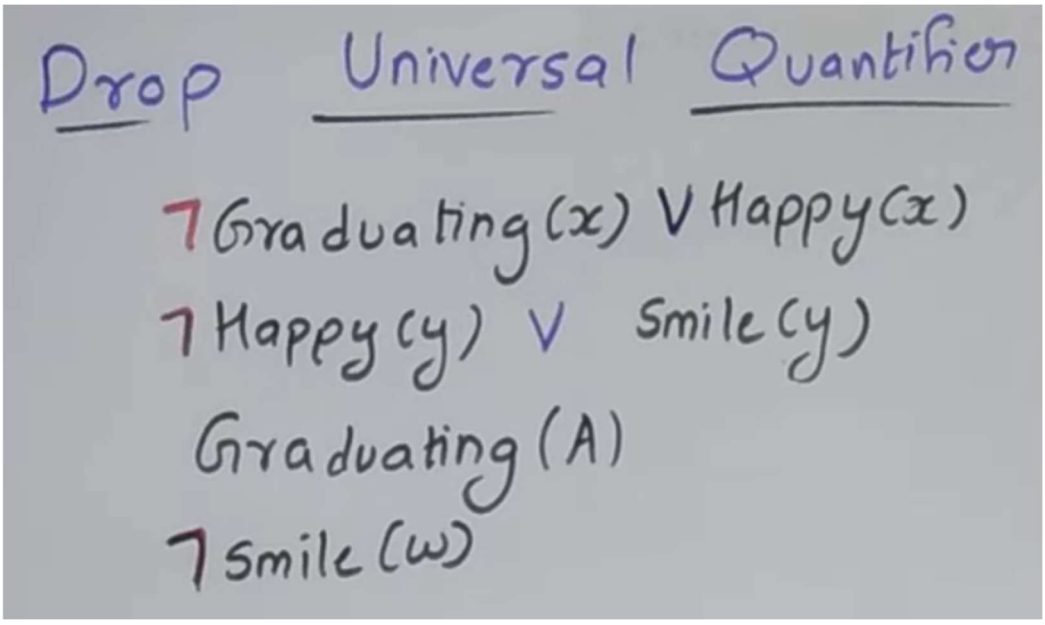
$\forall x [\neg \text{Graduating}(x) \vee \text{Happy}(x)]$

$\forall y [\neg \text{Happy}(y) \vee \text{Smile}(y)]$

$\exists z \ \text{Graduating}(z)$

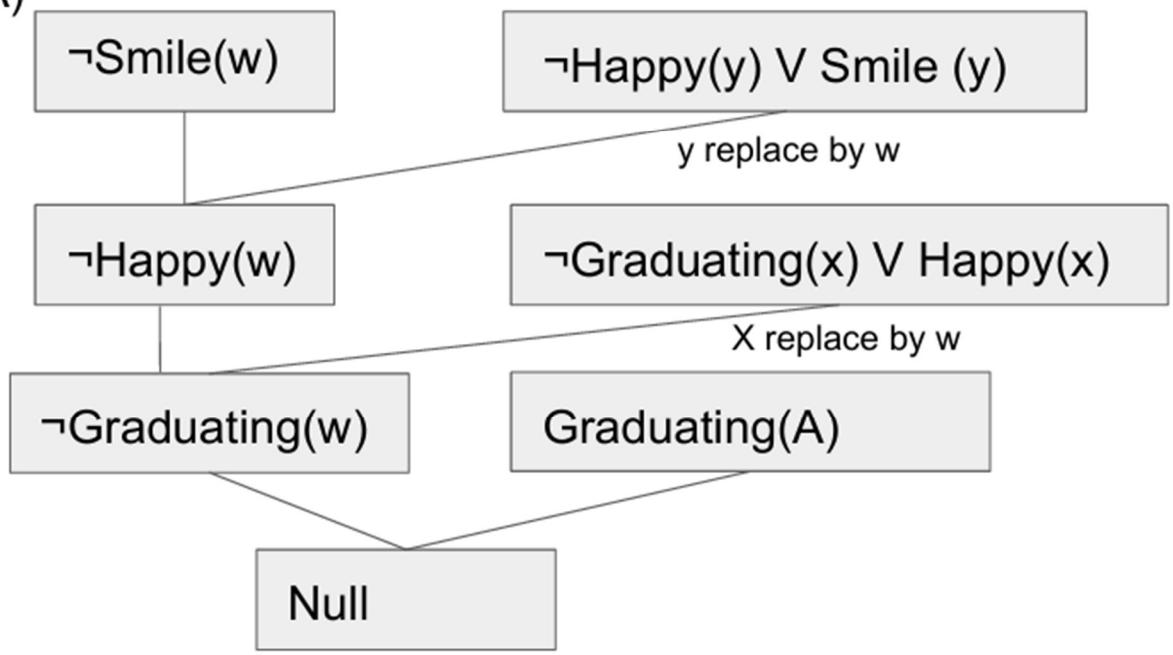
$\forall w \ \neg \text{smile}(w)$





Step 3: Draw Resolution tree

¬ Graduating (x) V Happy (x)
¬ Happy (y) V Smile (y)
Graduating (A)
¬ Smile (w)



Hence someone is smiling.

23. Explain different quantifiers with example.

1. Universal Quantifier (∀) — "For all"

The universal quantifier expresses that a certain statement or condition holds true for all elements in a given domain of discourse.

Syntax:

∀x P(x)

It means “For all x, P(x) is true.”

Example:

∀x (Bird(x) → Can Fly(x))

It stands for "All birds can fly."

2. Existential Quantifier (∃) — "There exists"

The existential quantifier indicates that at least one element in the domain satisfies the given predicate.

Syntax:

∃x P(x)

It means: “There exists an x such that P(x) is true.”

Example:

∃x (Student(x) ∧ Studies(x, AI))

It stands for "There exists a student who studies AI."

24. What do you mean by Resolution? Also discuss the steps in Resolution.

Resolution is a rule of inference used primarily in automated theorem proving and propositional/predicate logic. It is a method to prove whether a given statement (goal) logically follows from a set of known facts and rules (knowledge base).

Steps in Resolution (Propositional Logic):

Step 1: Convert all statements into CNF (Conjunctive Normal Form)

- CNF is a format where a formula is a conjunction (AND) of clauses, and each clause is a disjunction (OR) of literals.
- Example:
    - $A \rightarrow B$ becomes $\neg A \vee B$

Step 2: Negate the goal

- To prove a goal G, assume ¬G (not G) is true, and then attempt to derive a contradiction.
- This is called proof by contradiction.

Step 3: Add the negated goal to the knowledge base

- Combine all CNF statements (original knowledge base + ¬goal).

Step 4: Apply resolution repeatedly

- Choose two clauses that contain complementary literals (like A and ¬A).
- Resolve them by eliminating the complementary pair and forming a new clause.

Step 5: Continue until:

- You derive the empty clause (∅) – which means a contradiction has been found, so the original goal is proved true,
  OR
- No more new clauses can be generated – the goal is not provable from the given knowledge.

25. Define Belief network. Describe the steps of constructing belief network with an example.

Definition:

A Belief Network, also called a Bayesian Network, is a graphical model that represents a set of variables and their conditional dependencies using a directed acyclic graph (DAG).

- **Nodes** represent random variables.
- **Edges (arrows)** represent direct dependencies between variables.
- Each node is associated with a **Conditional Probability Table (CPT)** that quantifies the effects of the parent nodes.

Steps to Construct a Belief Network with Example of Medical Diagnosis (Cold and Fever):

Scenario:

We want to model how cold and flu can lead to fever, and how fever can cause shivering.

Step 1: Identify Relevant Variables

- Cold
- Flu
- Fever
- Shivering

Step 2: Determine Dependencies

- Cold and Flu cause Fever
- Fever causes Shivering

Step 3: Construct the Directed Acyclic Graph (DAG):

Step 4: Define Conditional Probability Tables (CPTs)

- P(Cold) = 0.3
- P(Flu) = 0.2
- P(Fever | Cold, Flu)
    - P(Fever = true | Cold = true, Flu = true) = 0.9
    - P(Fever = true | Cold = true, Flu = false) = 0.6
- P(Shivering | Fever)
    - P(Shivering = true | Fever = true) = 0.8
    - P(Shivering = false | Fever = false) = 0.9

Step 5: Validate the Network

1. **Check for Cycles**
    - The graph is acyclic (no loops exist).
    - All arrows flow in one direction:
      Cold & Flu → Fever → Shivering
2. **Check Completeness of CPTs**
    - Each node has an associated Conditional Probability Table (CPT):
        - Fever: Depends on Cold and Flu
        - Shivering: Depends on Fever
3. **Ensure Probabilities Sum to 1** in each CPT
    - For example, in P(Fever | Cold, Flu):
        - If Cold = true, Flu = false:
            - P(Fever = true) = 0.6
            - P(Fever = false) = 0.4 (i.e., sum = 1)

26. Explain various methods of knowledge representation.

1. Logical Representation

Logical representation uses symbols and logic to represent facts and rules about the world. The two most common types are:

- **Propositional Logic** – Deals with simple, declarative propositions
- **Predicate Logic (First-Order Logic)** – Represents objects, properties, and relations between them

Example:

- "All humans are mortal": ∀x (Human(x) → Mortal(x))
- "Socrates is a human": Human(Socrates)
- Therefore: Mortal(Socrates)

This type of representation is precise and supports formal reasoning and inference, but it can become complex for real-world applications with uncertainty.

2. Semantic Network

A semantic network is a graph-based structure where nodes represent concepts (objects, ideas), and edges represent relationships between them (such as "is-a", "has-a", "part-of").

Example:

Imagine the concept of a Penguin:

[Penguin] — is-a —> [Bird]

[Penguin] — can —> [Swim]

This network helps the system understand relationships and can be expanded easily by adding more nodes and edges.

Advantages:

- Easy to understand
- Good for hierarchical knowledge

3. Frames

Frames are data structures for representing stereotyped situations. Each frame has a name and consists of slots (attributes) and fillers (values). It resembles an object in object-oriented programming.

Example: Frame for a 'Car'

Frame: Car

Make: Toyota

Model: Corolla

Colour: Red

This representation makes it easy for the AI to answer questions like:

- What colour is the car? → Red

Advantages:

- Easy to modify
- Efficient for storing structured data

4. Production Rules

Production rules consist of IF-THEN statements that encode procedural knowledge. The rules are checked against facts in the knowledge base, and the appropriate rule is fired when conditions match.

Example:

- Rule: IF the sky is dark THEN switch on the light

The system uses these rules to infer what actions to take based on current conditions.

Advantages:

- Highly modular
- Easy to add/remove rules

5. Ontologies

An ontology is a formal representation of knowledge as a set of concepts within a domain and the relationships between them. It includes:

- Classes (concepts)
- Properties (relationships and attributes)
- Instances (individual examples)

Example (Healthcare Domain):

Class: Disease

Subclass: InfectiousDisease

Properties:

hasSymptom → Fever

Instance: Flu

Ontologies allow AI to understand domain-specific terms and enable semantic searching.

Advantage:

- Excellent for representing structured, domain-specific knowledge

27. Compare and contrast propositional logic and first order logic.

Parameter	Propositional Logic	First Order Logic (FOL)
1. Basic Unit	Propositions (whole statements)	Predicates with variables
2. Expressiveness	Less expressive	More expressive (can represent complex facts)
3. Variables	No variables	Uses variables to refer to objects
4. Objects and Relations	Cannot represent relations between objects	Can represent objects and their relationships
5. Quantifiers	Not supported	Uses quantifiers like ∀ (for all), ∃ (there exists)
6. Inference Capability	Limited inference; based on whole statements	More powerful inference using structure and relationships
7. Use Case	Suitable for simple, flat facts	Suitable for real-world scenarios and complex reasoning
8. Example	P = "It is raining"	Raining(x) → Wet(x) (If it's raining in place x, it's wet)

28. Write a short note on conditional probability and its role in AI.

Conditional probability is the likelihood of an event occurring given that another event has already happened. It is written as:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

In AI, conditional probability is essential for making predictions and decisions under uncertainty. It plays a key role in:

- **Bayesian Networks**: Modeling dependencies between variables.
- **Machine Learning**: Algorithms like Naive Bayes use it for classification.
- **NLP**: Predicts next words based on previous ones.
- **Decision Making**: Supports reasoning in uncertain environments.

Example: Conditional Probability in Spam Detection

- The probability that any email is spam: P(Spam)=0.3
- The probability that any email contains the word "free": P(Free)=0.25
- The probability that a spam email contains the word "free": P(Free|Spam)=0.7

$$P(Spam|Free) = \frac{P(Free|Spam) \cdot P(Spam)}{P(Free)} = \frac{0.7 \times 0.3}{0.25} = \frac{0.21}{0.25} = 0.84$$

If an email contains the word "free," there's an **84%** chance that it is spam.
This type of reasoning helps AI spam filters identify suspicious emails based on keywords

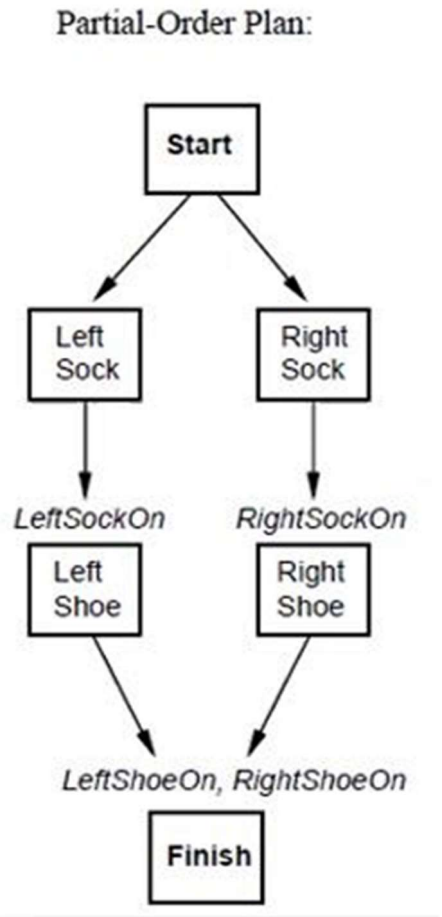
5. Planning and Learning

29. What is planning in AI? Explain Partial-order planning with suitable example.

Planning in AI is the process of thinking ahead and deciding a sequence of actions that an intelligent agent must perform to achieve specific goals. It involves generating a plan that transforms the initial state into a goal state using available actions and considering constraints.

Partial Order Planning does not enforce a strict order unless necessary. It allows actions to be executed in any sequence, as long as dependencies are respected. This means:

- Only actions that have dependencies are ordered.
- The planner commits to the minimum ordering constraints necessary.
- It allows flexibility in execution and parallelism.



Example:

The only requirement is that socks must be worn before shoes.

Socks can be worn simultaneously, as can shoes (after socks).

The agent can choose any valid sequence that respects constraints (e.g., left sock → right sock → left shoe → right shoe, or right sock → left sock → right shoe → left shoe, etc.).

Advantages:

1. More flexible, allows reordering.
2. Efficient when multiple tasks can run in parallel.

Disadvantages:

1. More complex to implement.
2. Requires additional reasoning for dependencies.

30. Problem on planning, PYQs:

i. Design a planning problem using STRIP for Air cargo transport. It involves loading and unloading cargo onto and off of planes and flying it from place.
Initial state: At SFO airport, Cargo1, Plane1 and at JFK airport, Cargo2, Plane2 is present.
Goal state: At SFO airport Cargo2 and at JFK airport Cargo1 is present. (asked twice)

i. STRIPS Representation – Air Cargo Transport Planning Problem

Problem Description:

This problem involves transporting cargo from one airport to another using planes. The task is to load and unload cargo onto planes and fly them between two airports.

STRIPS Representation:

Initial State:

At(Cargo1, SFO)

At(Plane1, SFO)

At(Cargo2, JFK)

At(Plane2, JFK)

Goal State:

At(Cargo1, JFK)

At(Cargo2, SFO)

Predicates:

1. At(X, Y) – Entity X is at location Y.
2. Cargo(X) – X is a cargo.
3. Plane(X) – X is a plane.
4. Loaded(X, Y) – Cargo X is loaded onto plane Y.
5. Flying(X, Y) – Plane X is flying to location Y.

Actions:

1. **Load Cargo onto Plane:**
   - **Preconditions:** At(Cargo, Airport) and At(Plane, Airport)
   - **Effects:** Loaded(Cargo, Plane) and Not(At(Cargo, Airport))

   Action: Load(Cargo, Plane, Airport)
2. **Unload Cargo from Plane:**
   - **Preconditions:** Loaded(Cargo, Plane) and At(Plane, Airport)
   - **Effects**: At(Cargo, Airport) and Not(Loaded(Cargo, Plane))

Action: Unload(Cargo, Plane, Airport)

3. **Fly Plane:**
    - **Preconditions:** At(Plane, Airport) and Loaded(Cargo, Plane)
    - **Effects:** At(Plane, DestinationAirport), At(Cargo, DestinationAirport), and Not(At(Plane, Airport))

    Action: Fly(Plane, FromAirport, ToAirport)

Solution Plan:

1. **Load Cargo1 onto Plane1 at SFO**:
    - Action: Load(Cargo1, Plane1, SFO)
2. **Fly Plane1 from SFO to JFK**:
    - Action: Fly(Plane1, SFO, JFK)
3. **Unload Cargo1 at JFK**:
    - Action: Unload(Cargo1, Plane1, JFK)
4. **Load Cargo2 onto Plane2 at JFK**:
    - Action: Load(Cargo2, Plane2, JFK)
5. **Fly Plane2 from JFK to SFO**:
    - Action: Fly(Plane2, JFK, SFO)
6. **Unload Cargo2 at SFO**:
    - Action: Unload(Cargo2, Plane2, SFO)

At the end of these actions, the goal state will be achieved: Cargo1 will be at JFK and Cargo2 will be at SFO.

ii. Consider problem of changing a flat tire. The goal is to have a good spare tire properly mounted on to the car's axle, where the initial state has a flat tire on the axle and a good spare tire in the trunk. Give the ADL description for the problem and also discuss the solution.

ii. Changing a Flat Tire Problem Using ADL

Problem Description:

The goal is to change a flat tire on a car, where the initial state has a flat tire on the car's axle, and a good spare tire in the trunk. The goal is to mount the good spare tire on the axle.

Initial State:

- OnAxle(FlatTire)
- InTrunk(SpareTire)

Goal State:

- OnAxle(SpareTire)

Predicates:

- OnAxle(tire) – Tire is on the axle
- InTrunk(tire) – Tire is in the trunk

Actions:

1. Remove Flat Tire

Action: RemoveFlatTire

Preconditions: OnAxle(FlatTire)

Effects: ¬OnAxle(FlatTire)

2. Take Spare Tire from Trunk

Action: TakeSpareFromTrunk

Preconditions: InTrunk(SpareTire)

Effects: ¬InTrunk(SpareTire)

3. Mount Spare Tire

Action: MountSpareTire

Preconditions: ¬OnAxle(FlatTire) ∧ ¬InTrunk(SpareTire)

Effects: OnAxle(SpareTire)

This sequence ensures the flat tire is removed and replaced by a good spare tire, achieving the goal state.

31. Explain the concept of PAC learning.

PAC learning is a theoretical framework that addresses the question of how much data is necessary for a learning algorithm to perform well on new, unseen data. The core idea is that a learning algorithm can be considered PAC if, given a sufficient number of training samples, it can produce a hypothesis that is likely (with high probability) to be approximately correct (within a specified error margin).

Given example in the form (X, f(X)) where:

- X is a binary input.
- f(X) is the output (TRUE/FALSE or 1/0),

The goal is to learn a function f: $\{0,1\}^n \rightarrow \{0,1\}$ using a hypothesis h from a hypothesis space H.

PAC Criterion:

The algorithm is PAC if:

$$P(|R(h) - \hat{R}(h)| \leq \epsilon) \geq 1 - \delta$$

Where:

- $R(h)$: true error of hypothesis h,
- $\hat{R}(h)$: error on training data,
- ϵ: allowable error (accuracy),
- δ: probability of failure (confidence = $1 - \delta$).

A learner is consistent if, for every ϵ and δ, there exists a sample size N such that:

$$P(|R(h) - \hat{R}(h)| > \epsilon) < \delta$$

Sample Complexity:

Minimum number of training examples N needed for PAC condition to hold.

If N is a polynomial function of $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$, then the hypothesis space H is **PAC-learnable**.

Advantages of PAC Learning:

- **Theoretical clarity:** Gives a formal understanding of learning processes.
- **Applies to many algorithms**: Supports decision-making in model selection.

Disadvantages:

- **Computational Complexity:** Finding the optimal hypothesis can be computationally expensive, especially as the hypothesis space grows.
- **Overfitting:** As the complexity of the hypothesis space increases, there is a risk of overfitting, where the model performs well on training data but poorly on unseen data.

Applications of PAC Learning

1. **Classification**:
   Helps design classifiers (e.g., decision trees, SVMs) that generalize well from limited labelled data.
2. **Active Learning**:
   PAC principles guide selection of the most informative samples, reducing labelling costs and improving efficiency.

32. Explain Reinforcement learning in detail.

Reinforcement Learning (RL) is a type of machine learning where an agent learns to make decisions by performing actions and receiving rewards or punishments from the environment. The goal is to learn the best strategy (called a policy) to maximize the total reward over time.

Working:

Reinforcement learning is based on trial and error. The agent:

1. **Observes the current state** of the environment.
2. **Takes an action** based on a policy.
3. **Receives a reward or penalty** from the environment.
4. **Updates its policy** to improve future actions.

Example: Robot Navigation

- Agent: Robot
- Goal: Reach destination without hitting obstacles
- Reward: +10 for reaching goal, -1 for each step, -100 for hitting a wall

Types of Reinforcement Learning:

1. **Model-Free**: Learns purely from experience.
2. **Model-Based**: Tries to understand how the environment works.

Advantages:

1. **Learns from Experience**: Improves performance over time by learning from trial and error.
2. **Works Without Labelled Data**: It doesn't need labelled input/output pairs

Applications:

1. Game playing (e.g., Chess)
2. Robotics

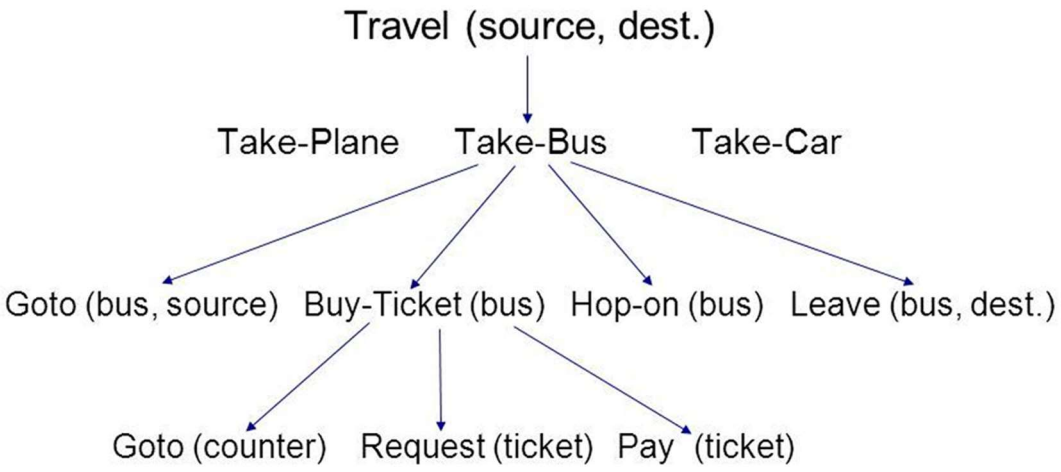
Disadvantages:

1. **Takes Time to Learn:** Often needs many trials before it learns the best actions, which can be slow.
2. **Requires Lots of Data:** Usually needs a large number of interactions with the environment to learn effectively.

33. Explain hierarchical planning in detail. #

Hierarchical Planning is a planning approach in AI where complex tasks are broken down into smaller, more manageable sub-tasks using a top-down strategy. This method simplifies decision-making by handling abstract goals first, then refining them into concrete actions.

Example:



Main Goal: Travel (source, dest.)
Decomposition into Subtasks:

i. Take-Plane
ii. Take-Bus
iii. Take-Car

Each of these methods represents a compound task that needs further breakdown.

Further Decomposition of "Take-Bus":

To Take-Bus, you need to:

i. Goto (bus, source)
ii. Buy-Ticket (bus)
iii. Hop-on (bus)
iv. Leave (bus, dest.)

Advantages of Hierarchical Planning:

1. **Reduces complexity** by focusing on high-level decisions first.
2. **Supports reusability** of task methods.
3. **More efficient** than flat planning in large domains.

Applications:

1. **Robotics (**e.g., robot navigation with multiple goals)
2. **Game AI** (e.g., character behaviour)

Disadvantages:

1. **Rigid structure**: Difficult to introduce new layers.
2. **Poor fault tolerance**

34. Explain the concept of Conditional order planning.

Conditional Order Planning (also known as contingency planning) is a planning strategy in Artificial Intelligence where the plan includes branches or conditions to handle situations where the outcome of actions is uncertain or depends on external factors.

Key Features:

- Plans contain branches for different outcomes.
- Allows flexibility in uncertain environments.
- Often used in robotics, navigation, and real-world AI systems.

Example:

Imagine a robot wants to enter a room:

- It tries to open the door.

Now, two possibilities:

- **Door is unlocked** → Enter the room.
- **Door is locked** → Use key or find another way.

35. Compare the importance of Partial order planning over Total order planning. #

Feature	Partial Order Planning (POP)	Total Order Planning (TOP)
Definition	Plans actions with only necessary order constraints	Plans actions in a fixed, complete sequence
Flexibility	High — allows actions to happen in any order if independent	Low — fixes the order of all actions even if not required
Handling Dependencies	Orders only dependent actions, leaving others unordered	Orders all actions strictly, regardless of dependencies
Concurrency	Supports parallel or concurrent actions	Does not support parallelism
Efficiency	More efficient in complex tasks by avoiding unnecessary ordering	Less efficient due to strict ordering
Implementation	More complex due to managing dependencies	Simpler to implement
Use Case	Suitable for dynamic, real-world tasks like robotics and workflows	Best for simple, linear tasks
Example	Pour coffee and toast bread in any order or simultaneously	Must pour coffee before toasting bread

36. What data is used to evaluate award and punishment of robot navigation.

In robot navigation, the data used to evaluate reward (award) and punishment typically comes from the environment and is part of a reinforcement learning setup.

Reinforcement learning involves an agent that interacts with an environment and learns optimal actions through rewards and penalties. The agent continuously improves its strategy by maximizing long-term rewards.

Data Used to Evaluate Rewards and Punishments:

1. **Distance to Goal**
    - Reward: If the robot moves closer to the goal.
    - Punishment: If the robot moves away from the goal.
2. **Collision Detection**
    - Punishment: If the robot hits an obstacle or wall.
3. **Time Taken**
    - Punishment: For taking too long to reach the goal.
    - Reward: For reaching the goal quickly.

Example:

If a robot is navigating a maze:

- It gets **+10** for reaching the goal,
- **-5** for hitting a wall,
- **+1** for each step toward the goal

37. Explain the concept of Supervised learning.

Supervised learning is a machine learning approach where the model is trained on labelled data, meaning each input has a corresponding correct output. The model learns from this data to make predictions or classifications for new, unseen inputs. It is commonly used in tasks where historical data with known outcomes is available.

Key Points:

- Requires labelled data.
- Accurate labels are crucial.
- Performs better with more data.

Working:

1. **Training Data**: You provide the algorithm with input-output pairs.
    - Example:
        - Input: Email text Output: “Spam” or “Not Spam”
2. **Model Learns**: The algorithm learns the relationship between inputs and outputs.
3. **Prediction**: Once trained, it can predict outputs for new, unseen inputs.

Example:

A spam email classifier is trained using emails labelled as "spam" or "not spam."

6. AI Applications

38. Explain different applications of AI in Robotics, Healthcare, Retail and Banking.

1. AI in Robotics

AI enables robots to perform tasks that require reasoning, learning, and decision-making.

Applications:

- **Autonomous Robots:** Self-driving delivery robots and drones that navigate environments.
- **Industrial Automation:** Robotic arms using AI for precision assembly, welding, or quality inspection.
- **Humanoid Robots:** Robots like Sophia that interact using natural language and facial recognition.
- **Cleaning Robots:** Smart vacuum cleaners (e.g., Roomba) using AI to map rooms and clean efficiently.

2. AI in Healthcare

AI is transforming diagnosis, treatment, and patient care.

Applications:

- **Medical Imaging:** AI analyzes X-rays, MRIs to detect tumours or fractures.
- **Diagnosis Support:** Tools like IBM Watson assist doctors by suggesting diagnoses based on patient data.
- **Predictive Analytics:** AI predicts disease outbreaks or patient readmissions.
- **Robotic Surgery:** AI-powered surgical robots assist with precision operations.

3. AI in Retail

AI helps improve customer experience, inventory management, and sales forecasting.

Applications:

- **Recommendation Systems:** Suggest products based on user behaviour (used by Amazon, Flipkart).
- **Chatbots:** Virtual assistants help customers with queries 24/7.
- **Demand Forecasting:** AI predicts which products will be in demand.
- **Visual Search:** Customers upload a photo, and AI finds similar products.

4. AI in Banking

AI enhances fraud detection, customer service, and financial forecasting.

Applications:

- **Fraud Detection:** AI monitors transactions for unusual behaviour patterns.
- **Credit Scoring:** AI evaluates loan applications more accurately by analysing more variables.
- **Chatbots & Virtual Assistants:** Assist customers with banking queries (e.g., HDFC's EVA, SBI's SIA).
- **Algorithmic Trading:** AI analyzes market trends and executes trades at optimal times.

39. Write detailed note on: Natural Language Processing / Language models of NLP.

Definition:

Natural Language Processing (NLP) is a subfield of Artificial Intelligence and Computational Linguistics that focuses on the interaction between computers and human languages. The goal of NLP is to enable machines to understand, interpret, and generate human language in a way that is both meaningful and useful.

Steps involved in NLP (2022 May):

Lexical analysis – Breaks text into words or tokens.
It identifies meaningful units like keywords, punctuation, etc.

Syntactic analysis – Checks grammar and sentence structure.
It ensures the sentence follows language rules.

Semantic analysis – Determines the meaning of words and sentences.
It checks if the sentence makes logical sense.

Discourse integration – Connects meaning across multiple sentences.
It relates new information to what has already been said.

Pragmatic analysis – Understands context and intended meaning.
It interprets the speaker's true intent in real situations.

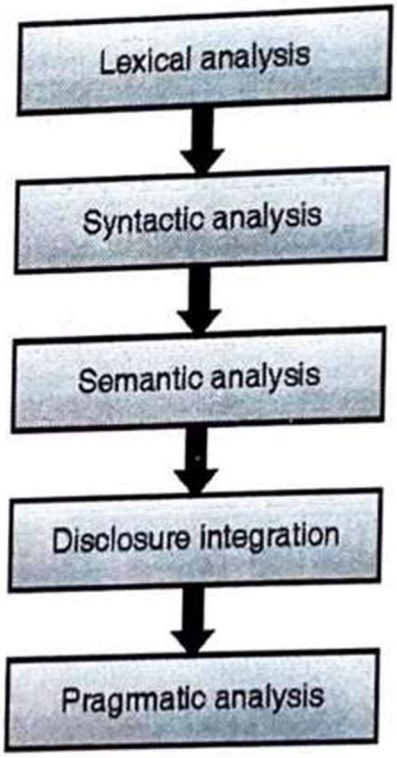


Fig. 6.2.1 : Steps of NLP process

Language Models in NLP (asked twice):

Language models in NLP are algorithms designed to understand, generate, and predict human language. They use statistical and machine learning techniques to process text data. These models can be broadly classified into:

1. **N-gram Models** – Predicts the next word based on the previous n words. Example: A bigram model considers two-word sequences.
2. **Hidden Markov Models (HMM)** – Uses probabilities to model sequences of words, often applied in speech recognition and part-of-speech tagging.
3. **Neural Network-Based Models** – Uses deep learning to understand language context, including:
    - **Recurrent Neural Networks (RNNs)** – Processes sequential data but struggles with long dependencies.
    - **Long Short-Term Memory (LSTM)** – A type of RNN designed to handle longer sequences.
    - **Transformers (e.g., BERT, GPT)** – Uses self-attention mechanisms to capture relationships between words across large text spans, enabling tasks like translation and summarization.

NLP Techniques

1. Pattern Matching

Pattern matching in NLP involves searching for specific word patterns or sequences in text using rules or regular expressions.

2. Syntactically Driven Parsing

This technique analyzes the grammatical structure of a sentence using syntax rules to understand relationships between words (like subject, verb, object).

Applications of NLP:

- **Machine Translation:** Converts text from one language to another (e.g., Google Translate).
- **Chatbots & Virtual Assistants:** Enables natural conversation with tools like Siri and Alexa.
- **Sentiment Analysis:** Detects emotions in reviews or social media posts (positive, negative, neutral).
- **Spam Detection:** Filters out unwanted or harmful emails.
- **Speech Recognition:** Converts spoken words into text (used in voice typing).

40. Give types of parsing and generate the parse tree for the sentence “The cat ate the fish.” #

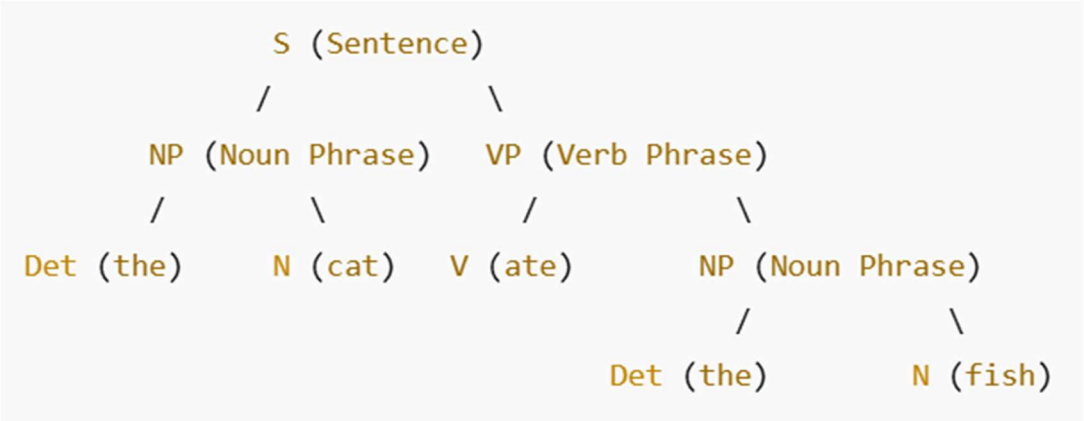
Parsing is the process of analyzing a sentence to understand its grammatical structure.

Types of Parsing:

1. **Top-Down Parsing:**
   - Starts from the start symbol (S) and breaks it down to match the input.
   - Follows the grammar rules to generate all possible sentences.
2. **Bottom-Up Parsing:**
   - Starts with the input and tries to reach the start symbol.
   - Useful in syntax analysis of compilers.

Parse Tree for the sentence: “The cat ate the fish.”

Using basic English grammar rules (Context-Free Grammar – CFG):



S (Sentence) is split into:

- **NP (The cat)** → A noun phrase with a determiner ("The") and a noun ("cat").
- **VP (ate the fish)** → A verb phrase with:
  - **V (ate)** → the verb
  - **NP (the fish)** → another noun phrase made of a determiner ("the") and a noun ("fish").

~ AJ