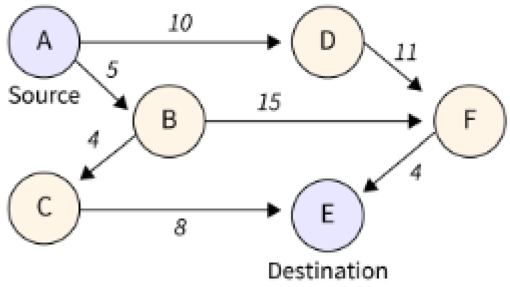
| USN | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|

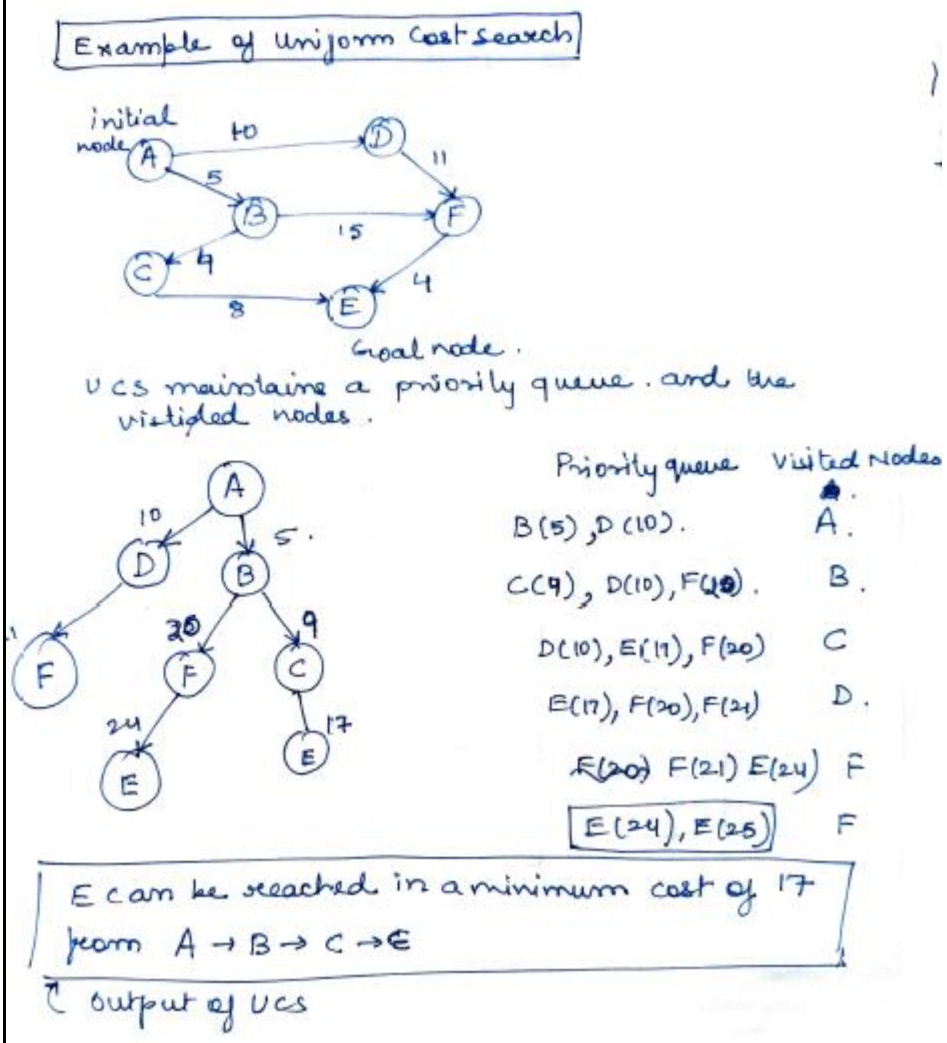
# Internal Assessment Test 1 – Sept. 2025

| Sub: | Intelligent Systems and Machine Learning Algorithms | | | | | | | Code: | BEC515A |
|---|---|---|---|---|---|---|---|---|---|
| Date: | 07/ 11 / 2024 | Duration: | 90 mins | Max Marks: | 50 | Sem: | V | Branch: | ECE |

| | Answer any 5 full questions | Marks | CO | RBT |
|---|---|---|---|---|
| 1 | a) What are the key differences between informed search and uninformed search strategies?<br><br>Sol:<br><br>Informed Search, also known as heuristic search, involves using specific knowledge or heuristics about a problem to find solutions more efficiently. In contrast, Uninformed Search, or blind search, does not have such additional information and explores the search space without guidance.<br><br>Informed Search is often faster and more efficient as it can intelligently narrow down the search space, thanks to its use of heuristics. Uninformed Search methods, on the other hand, systematically explore the search space and are typically slower due to lack of guidance.<br><br>Examples of Informed Search algorithms include A* and Greedy Best-First Search, which use heuristics to estimate the best path to a goal. Uninformed Search algorithms like Breadth-First Search and Depth-First Search explore all paths equally without such estimations.<br><br>Informed Search is ideal for complex problems where some understanding of the problem space is available. Uninformed Search is suitable for simpler problems or when little is known about the domain.<br><br>b) Explain Uniform-cost search with an example.<br><br>Sol: Uniform Cost Search (UCS) is a search algorithm used in artificial intelligence (AI) for finding the least cost path in a graph. It is a variant of Dijkstra's algorithm and is particularly useful when all edges of the graph have different weights, and the goal is to find the path with the minimum total cost from a start node to a goal node.<br><br>UCS uses a priority queue to store nodes. The node with the lowest cumulative cost is expanded first. This ensures that the search explores the most promising paths first. UCS calculates the cumulative cost from the start node to the current node and prioritizes nodes with lower costs. UCS explores nodes by expanding the least costly node first, continuing this process until the goal node is reached. The path to the goal node is guaranteed to be the least costly one.<br><br>Example: | [10] | CO3 | L2 |




We will have an empty priority queue and a boolean visited array. We will insert A with cost 0 into the queue.




2 What are the key elements of problem formulation, and how does problem formulation impact the effectiveness of a search strategy? Explain with example. [10] CO1 L1

Sol:

In **Artificial Intelligence (AI) problem-solving**, the first and most crucial step is **problem formulation**, because it defines *what exactly the agent needs to do*. A well-formulated problem allows the search algorithm to find an effective solution efficiently.

| **Element** | **Description** | **Example (8-Puzzle Problem)** |
|---|---|---|
| **1. Initial state** | The starting point of the problem — where the agent begins. | The initial configuration of tiles on the 3×3 puzzle board. |

| | | | | | | |
|---|---|---|---|---|---|---|
| | **2. Actions (Operators)** | The set of all possible actions that can be taken from a given state. | Moving the blank tile **up**, **down**, **left**, or **right** (when possible). | | | |
| | **3. Transition model** | Describes the result of each action — what new state results from taking a particular action in a state. | If the blank moves left, the tile left of it moves into its position, changing the board configuration. | | | |
| | **4. Goal state (or goal test)** | Defines what it means to have *solved* the problem. | When the tiles are in order (1–8) with the blank in the bottom-right corner. | | | |
| | **5. Path cost (or cost function)** | Assigns a numerical cost to each path — used to compare alternative solutions. | Usually, the cost of each move = 1; total cost = number of moves. | | | |

Example:

An agent needs to find the shortest path from **Arad** to **Bucharest** (a classic AI example).

- **Initial state:** Arad
- **Actions:** Drive to neighboring cities connected by roads.
- **Transition model:** Resulting city after traveling along a road.
- **Goal test:** City = Bucharest
- **Path cost:** Total distance (in km)

If the formulation is **clear**, a search algorithm (like Uniform Cost Search or A*) will find the *optimal path quickly* (Arad → Sibiu → Fagaras → Bucharest).

If the formulation **misses important details** (e.g., ignores distances or includes irrelevant roads), the search might explore **unnecessary paths**, increasing time and cost, or even fail to find the shortest route.

A well-formulated problem simplifies the search space and guides the search algorithm efficiently toward the goal, whereas a poor formulation can make even simple problems computationally expensive or unsolvable.

| | | | | |
|---|---|---|---|---|
| 3 | What is a "state space," and how is it used in the context of problem-solving? Give an example of a state space in Vacuum Cleaner search problem? | [10] | CO2 | L3 |

Sol:

A **state space** is the **set of all possible states** that can be reached by an agent when performing actions starting from the initial state.

In simple terms:

The **state space** represents *everything that could possibly happen* in the environment — every situation the agent might find itself in while trying to solve the problem. **state space** is the entire "map" of possible situations the agent can be in. The search algorithm navigates this map to find a path from the start (initial state) to the destination (goal state).

It defines the "search world" in which an AI agent looks for a solution.

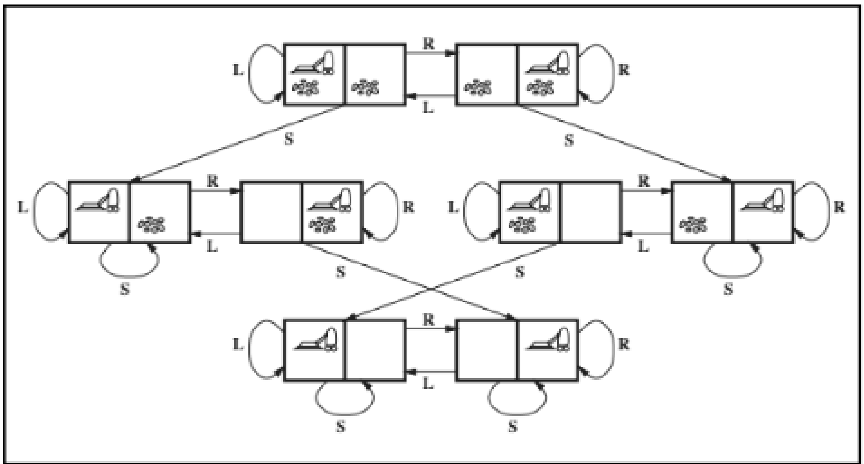



**Figure 3.3** The state space for the vacuum world. Links denote actions: L = *Left*, R = *Right*, S = *Suck*.

CI CCI HOD

| | | | | |
|---|---|---|---|---|
| 4 | What is an Intelligent agent? Explain in detail the structure of an intelligent agent and the interactions of the different components of an intelligent agent with neat diagram.<br>Sol:<br>An **Intelligent Agent (IA)** is an **autonomous system** that **perceives** its environment through sensors and **acts** upon that environment through actuators **to achieve specific goals**.<br>**Definition:**<br>An intelligent agent is an entity that perceives its environment and takes actions to maximize its chances of achieving its goals.<br>• For example: A human agent has eyes, ears, and other organs for sensors and hands, legs, vocal tract, and soon for actuators.<br>• A robotic agent might have cameras and infrared range finders for sensors and various motors for actuators.<br>• A software agent receives keystrokes, file contents, and network packets as sensory inputs and acts on the environment by displaying on the screen, writing files, and sending network packets. | [10] | CO1 | L1 |

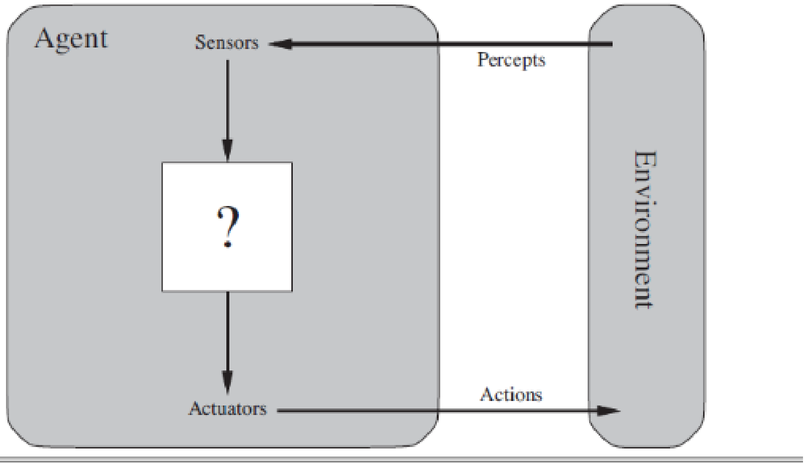



**Figure 2.1** Agents interact with environments through sensors and actuators.

**A. Sensors**

- Used to **perceive** or collect data from the environment.
- Example: camera, microphone, temperature sensor, etc.

**B. Actuators**

- Used to **act** or make changes in the environment.
- Example: wheels, motors, display, robotic arms, etc.

**C. Agent Program (the brain)**

- The decision-making component that processes perceptions and selects appropriate actions.
- It can include:
    - **Performance measure** (defines success)
    - **Knowledge base** (what the agent knows)
    - **Inference/Decision-making system**
    - **Learning module** (improves with experience)

**Cycle of Interaction:**

1. **Perception:** Sensors perceive the environment.
2. **Reasoning/Decision:** The agent processes inputs and decides the next action using its knowledge.
3. **Action:** Actuators execute the chosen action.
4. **Learning:** The agent evaluates outcomes and learns to improve future actions.

5 Explain different types of Agents. With examples. [10] CO1 L1

| Type | Description | Example |
|---|---|---|
| 1. **Simple Reflex Agent** | Acts only on the **current percept**. Ignores history or future consequences. Uses **condition–action rules** ("if–then" rules). | A vacuum cleaner that sucks dirt if it detects dirt; otherwise moves randomly. |
| 2. **Model-Based Reflex Agent** | Maintains an **internal model** of the world (keeps track of partially observable states). Uses both **current percept** and **stored information**. | A robot that remembers which rooms are already clean and which are dirty. |
| 3. **Goal-Based Agent** | Acts to achieve specific **goals**. Uses **search** and **planning** to find sequences of actions that reach the | A GPS navigation system that plans the shortest route from start to |

| | | |
|---|---|---|
| | goal. | destination. |
| 4. **Utility-Based Agent** | Chooses actions based on **utility (happiness, satisfaction)** — not just goal achievement but how *good* the outcome is. Handles **trade-offs** among multiple goals. | A self-driving car that not only reaches its destination but also minimizes fuel use and travel time. |
| 5. **Learning Agent** | Can **learn and improve** its performance from experience. Has a **learning component** that updates its knowledge and decision-making. | ChatGPT, AlphaGo, or a recommendation system that improves with user feedback. |

6 What is the PEAS framework, and how is it used to define the components of an agent? Provide PEAS descriptions of the following agents: a) Medical diagnosis system b) Part-picking Robot [10] CO1 L2

**PEAS** stands for:
**P – Performance measure**
**E – Environment**
**A – Actuators**
**S – Sensors**

**Purpose of PEAS:**

- To **clearly specify** what the agent needs to do, where it operates, and how it interacts with the environment.
- It helps in **designing agent programs** by breaking down the problem into understandable parts.

| Component | Description | Example (Vacuum Cleaner Agent) |
|---|---|---|
| **Performance Measure (P)** | Criteria that define how successful the agent is. | Amount of dirt cleaned, cleaning time, energy used. |
| **Environment (E)** | The external world the agent operates in. | Rooms with dirt, walls, obstacles. |
| **Actuators (A)** | Devices through which the agent acts on the environment. | Move left/right, suck dirt. |
| **Sensors (S)** | Devices through which the agent perceives the environment. | Dirt sensor, position sensor. |

| Agent Type | Performance Measure | Environment | Actuators | Sensors |
|---|---|---|---|---|
| **Medical Diagnosis System** | Diagnostic accuracy, patient recovery | Patients, medical databases | Display, alerts, records | Patient data, test results |
| **Part-Picking Robot** | Picking accuracy, speed, efficiency | Factory floor, parts bins | Robotic arm, motors | Cameras, proximity sensors |

7 a) What are the four main performance parameters (completeness, optimality, time complexity, and space complexity) used to evaluate search algorithms. [5] CO2 L2

[5] CO2 L2

Sol:

| Parameter | Definition | What It Measures | Example / Note |
|---|---|---|---|
| **1. Completeness** | Determines whether the algorithm is **guaranteed to find a solution** (if one exists). | *Reliability* of the search. | BFS is **complete** because it will eventually find a solution if one exists. DFS is **not complete** in infinite state spaces. |
| **2. Optimality** | Checks whether the algorithm **finds the best (least-cost or shortest)** solution among all possible ones. | *Quality* of the solution. | BFS is **optimal** if all step costs are equal. A* is **optimal** if the heuristic is admissible. |
| **3. Time Complexity** | Measures the **total time** required to find a solution (in terms of number of nodes generated or expanded). | *Speed / Efficiency* | Depends on the number of states explored; BFS has exponential time complexity in large spaces. |
| **4. Space Complexity** | Measures the **amount of memory** required during the search process (for storing nodes, paths, etc.). | *Memory usage / Scalability* | DFS uses less space than BFS because it stores only a single path. |

b) Explain Depth first search Strategy with functional description and performance measures.

Sol:

**Depth First Search (DFS)** is a **uninformed (blind) search strategy** that explores a search tree by **expanding the deepest unexpanded node first**.

It goes **as deep as possible** along one branch before **backtracking** to explore other branches.

The basic algorithm:

- Start from the **initial state (root node)**.
- Explore **one successor (child)** completely before moving to the next.
- When a **dead-end (no more successors)** is reached, **backtrack** to the most recent node that still has unexplored paths.
- Continue until a **goal state** is found or all nodes are explored.

1. Start with a stack containing the initial state.
2. Repeat until the stack is empty:
   a. Pop the top node from the stack (current node).
   b. If the node is a goal state → return success (solution found).
   c. Else, generate all successors of the node.
   d. Push the successors onto the stack (in any order).
3. If stack becomes empty → return failure (no solution found).

| Performance Parameter | Description (for DFS) |
|---|---|
| **Completeness** | *Not complete* in infinite-depth or cyclic spaces (it may get stuck going infinitely deep). *Complete* if the search space is finite. |
| **Optimality** | *Not optimal* — may find a suboptimal (longer) path before the shortest one. |
| **Time Complexity** | O(b^m) — where **b** = branching factor, **m** = maximum depth of the search tree. |
| **Space Complexity** | O(bm) — only stores one path from root to leaf (plus unexpanded siblings). Very space efficient. |

CI　　　　CCI　　　　HOD