INTRODUCTION TO AI AND APPLICATIONS

BETC105/205

Module 1

BOS (CS/IS)
VTU Belagavi

Prepared by

Dr. Likewin Thomas,
BOS Members (CS/IS) - VTU Belagavi
Prof. & Head, Dept of AIML,
PESITM Shivamogg

Dr. Pavan Kumar M P,
BOS Members (CS/IS) - VTU Belagavi
Assoc. Prof., Dept of ISE,
JNNCE Shivamogg

Dr. Demian Antony D'Mello,
BOS Chairman (CS/IS) - VTU Belagavi
Vice Principal and Dean - Academics
Canara Engineering College

Outline

Module 1: Introduction to Artificial Intelligence

Outline

Module 2: Introduction to Prompt Engineering

1. Introduction to Prompt Engineering

- Overview of Prompt Engineering
- The Evolution of Prompt Engineering
- Types of Prompts
- How Does Prompt Engineering Work?
- The Role of Prompt Engineering in Communication
- The Advantages of Prompt Engineering
- The Future of Large Language Model (LLM) Communication

2. Prompt Engineering Techniques for ChatGPT

- Introduction to Prompt Engineering Techniques
- Instructions Prompt Technique
- Zero, One, and Few Shot Prompting
- Self-Consistency Prompt

3. Prompts for Creative Thinking

- Introduction to Creative Thinking with Prompts
- Unlocking Imagination and Innovation

4. Prompts for Effective Writing

- Introduction to Writing with Prompts
- Igniting the Writing Process with Prompts

Outline

Module 3: Machine Learning

1. Machine Learning in AI
- Overview of Machine Learning Techniques
- Introduction to Machine Learning Models

2. Regression Analysis in Machine Learning
- Basics of Regression
- Linear and Non-Linear Regression Techniques

3. Classification Techniques
- Overview of Classification Algorithms
- Naïve Bayes Classification
- Support Vector Machine (SVM)

4. Clustering Techniques
- Introduction to Clustering
- Types of Clustering Algorithms

5. Neural Networks
- Basics of Neural Networks
- Types and Applications of Neural Networks

Outline

Module 4: Trends in AI

1. AI and Ethical Concerns

- Introduction to AI Ethics
- Ethical Implications in AI Development
- Addressing Bias and Fairness in AI

2. AI as a Service (AIaaS)

- Overview of AI as a Service
- Benefits and Challenges of AIaaS
- Popular AIaaS Platforms

3. Recent Trends in AI

- Overview of Current AI Trends
- Key Developments in AI Research and Applications

4. Expert Systems

- Introduction to Expert Systems
- Components of Expert Systems
- Applications of Expert Systems

5. Internet of Things (IoT)

- Introduction to IoT
- IoT Architecture and Components
- IoT Applications in Various Industries

6. Artificial Intelligence of Things (AIoT)

- AIoT: Combining AI and IoT
- Applications of AIoT in Smart Cities, Healthcare, and Industry 4.0

Outline

Module 5: Robotics and Industrial Applications of AI

1. Robotics: An Application of AI
- Introduction to Robotics
- Robotics and AI Integration
- Robotics as an Application of AI

2. Drones Using AI
- Introduction to AI in Drones
- AI Technologies Powering Drones
- Applications of AI in Drone Technology

3. No Code AI and Low Code AI
- Overview of No Code and Low Code AI
- Benefits and Use Cases of No Code AI
- Developing AI Solutions with Low Code Platforms

4. Industrial Applications of AI
- **AI in Healthcare**
  - Applications of AI in Healthcare Diagnostics and Treatment
- **AI in Finance**
  - AI in Risk Management, Fraud Detection, and Algorithmic Trading
- **AI in Retail**
  - AI in Inventory Management, Customer Personalization, and Sales Forecasting
- **AI in Agriculture**
  - Precision Agriculture and AI for Crop Management
- **AI in Education**
  - AI in Adaptive Learning, Content Recommendation, and Student Assessment
- **AI in Transportation**
  - Autonomous Vehicles and AI for Traffic Management
- **AI in Experimentation and Multi-disciplinary Research**
  - AI in Scientific Research and Innovation

INTRODUCTION TO AI AND APPLICATIONS

BETC105/205

Module 1

BOS (CS/IS)
VTU Belagavi

Prepared by

Dr. Likewin Thomas,
BOS Members (CS/IS) - VTU Belagavi
Prof. & Head, Dept of AIML,
PESITM Shivamogg

Dr. Pavan Kumar M P,
BOS Members (CS/IS) - VTU Belagavi
Assoc. Prof., Dept of ISE,
JNNCE Shivamogg

Dr. Demian Antony D'Mello,
BOS Chairman (CS/IS) - VTU Belagavi
Vice Principal and Dean - Academics
Canara Engineering College

Outline

Module 1: Introduction to Artificial Intelligence

1. Introduction to Artificial Intelligence

1. Definition of Artificial Intelligence
2. How Does AI Work?
3. Advantages and Disadvantages of Artificial Intelligence
4. History of Artificial Intelligence
5. Types of Artificial Intelligence:
    a. Weak AI vs. Strong AI
    b. Reactive Machines
    c. Limited Memory
    d. Theory of Mind
    e. Self-Awareness
6. Is Artificial Intelligence the Same as Augmented Intelligence and Cognitive Computing?
7. Introduction to Machine Learning and Deep Learning

2. Machine Intelligence

1. Defining Intelligence
2. Components of Intelligence
3. Differences Between Human and Machine Intelligence
4. Agent and Environment in AI
5. Search Algorithms:
    a. Uninformed Search Algorithms
    b. Informed Search Algorithms:
        i. Pure Heuristic Search
        ii. Best-First Search Algorithm (Greedy Search)

Definition of Artificial Intelligence

- **Artificial Intelligence (AI)** is the science and engineering of making intelligent machines, especially intelligent computer programs, *John McCarthy (2004).*
- **Historical Background:** The concept of AI was first explored in 1950 by Alan Turing, a British mathematician and computer scientist, who asked the question "Can machines think?" This was a groundbreaking idea and led him to propose the Turing Test.
- **The Role of John McCarthy (2004):** Later, in 2004, John McCarthy defined AI as the science and engineering of making intelligent machines—basically, how we can program computers to act smart like humans.
- **A Researcher's Perspective:** For researchers, AI refers to a set of algorithms (step-by-step instructions) that help a machine make decisions and act without being explicitly told what to do each time.
- **Understanding AI from a Simple View:** Think of AI as machines or software that are designed to learn from their environment, just like humans learn from their experiences. For example, an AI program can be trained to recognize your face.
- **Everyday AI Examples:** Some well-known examples of AI include chess-playing computers or self-driving cars. These systems depend on deep learning (a type of AI that mimics the human brain) and natural language processing (helping computers understand human language, like Siri or Google Assistant).

Module 1: Introduction to Artificial Intelligence

How Does AI Work?

- AI works by processing large datasets, recognizing patterns, and making decisions using algorithms.
- It involves learning, reasoning, and self-correction:
  a. Learning - AI learns from data
  b. Reasoning - AI chooses the correct algorithm
  c. Self-Correction - AI refines algorithms for accuracy

Advantages and Disadvantages of AI

Advantages:

1. Performs well on tasks that uses detailed data.
2. Takes less time to perform tasks that needs to process huge volumes of data.
3. Generates consistent and accurate results.
4. Can be used 24 X 7.
5. Optimizes tasks by better utilizing resources.
6. Automates complex processes.
7. Minimizes downtime by predicting maintenance needs.
8. Enables companies to produce new products having better quality and speed.

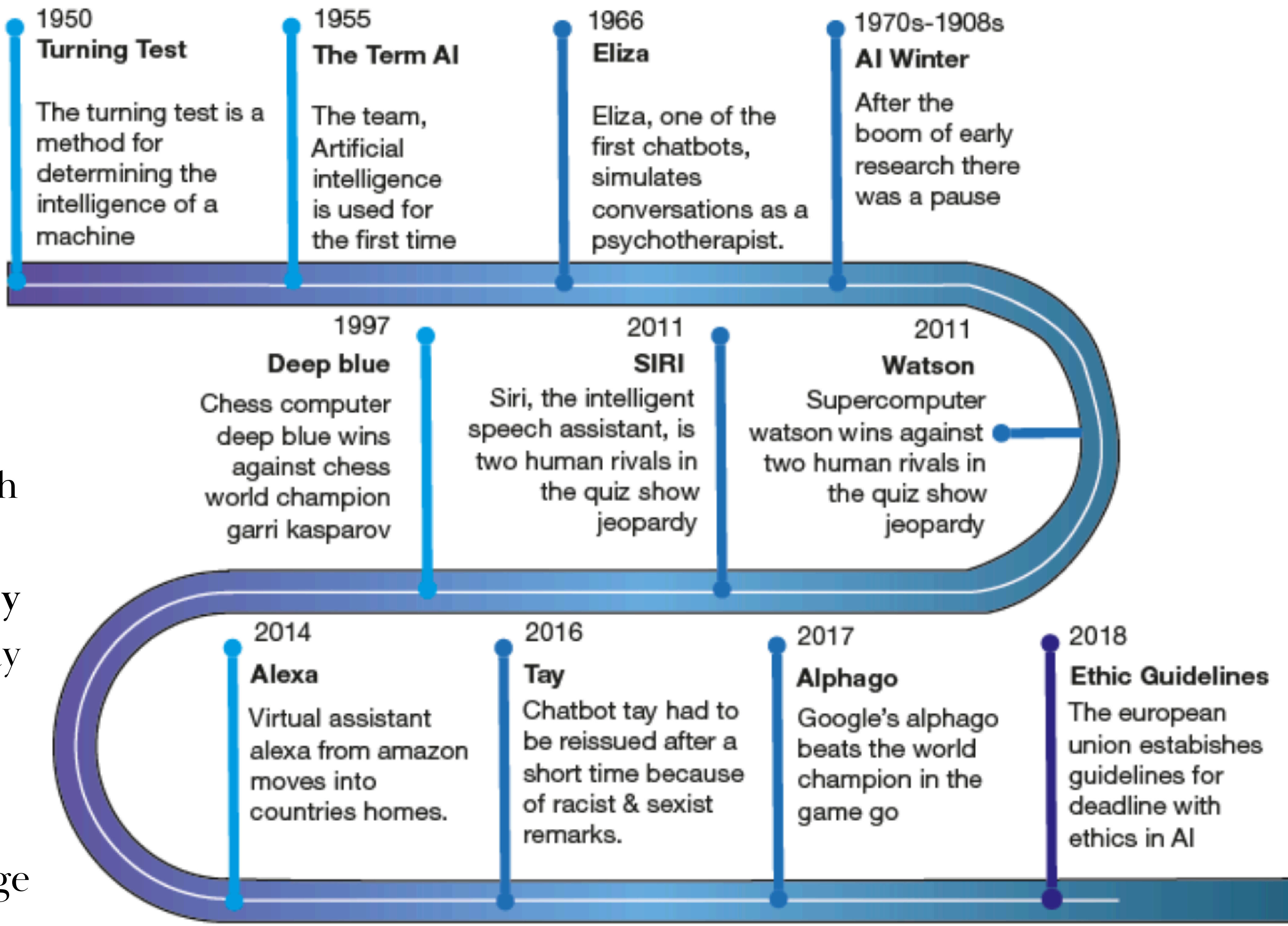
Disadvantages:

1. Involves more cost.
2. Technical expertise required to develop and use AI applications.
3. Lack of trained professionals.
4. Incomplete or inaccurate data may result in disastrous results.
5. Lacks the capability to generalize tasks

Module 1: Introduction to Artificial Intelligence

History of AI

- **1943 - First Neural Network Model Proposed:** By Warren McCullough and Walter Pitts to lay the foundation for artificial neural networks and machine learning.
- **1950 - Turing Test Introduced:** By Alan Turing to measure a machine's ability to exhibit intelligent behavior indistinguishable from humans.
- **1956 - John McCarthy Coins the Term 'Artificial Intelligence':** By John McCarthy during the Dartmouth Conference, marking the birth of AI as a formal field.
- **1997 - IBM Deep Blue Defeats Chess Champion Garry Kasparov:** By IBM's team to demonstrate AI's capability in strategic decision-making through computational power.
- **2011 - Siri Introduced by Apple:** By Apple to revolutionize personal assistants using natural language processing and AI for everyday tasks.

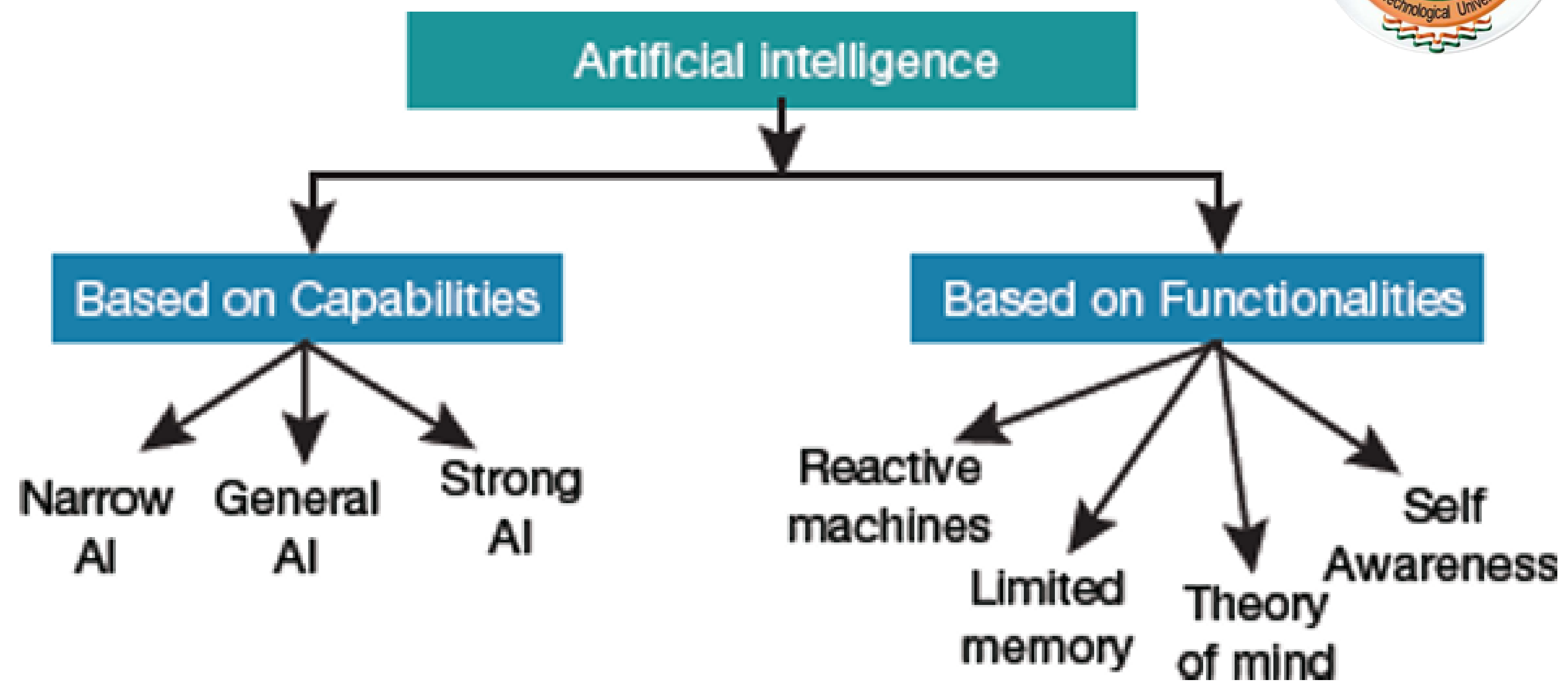


Module 1: Introduction to Artificial Intelligence

Types of AI

AI can be categorized into:

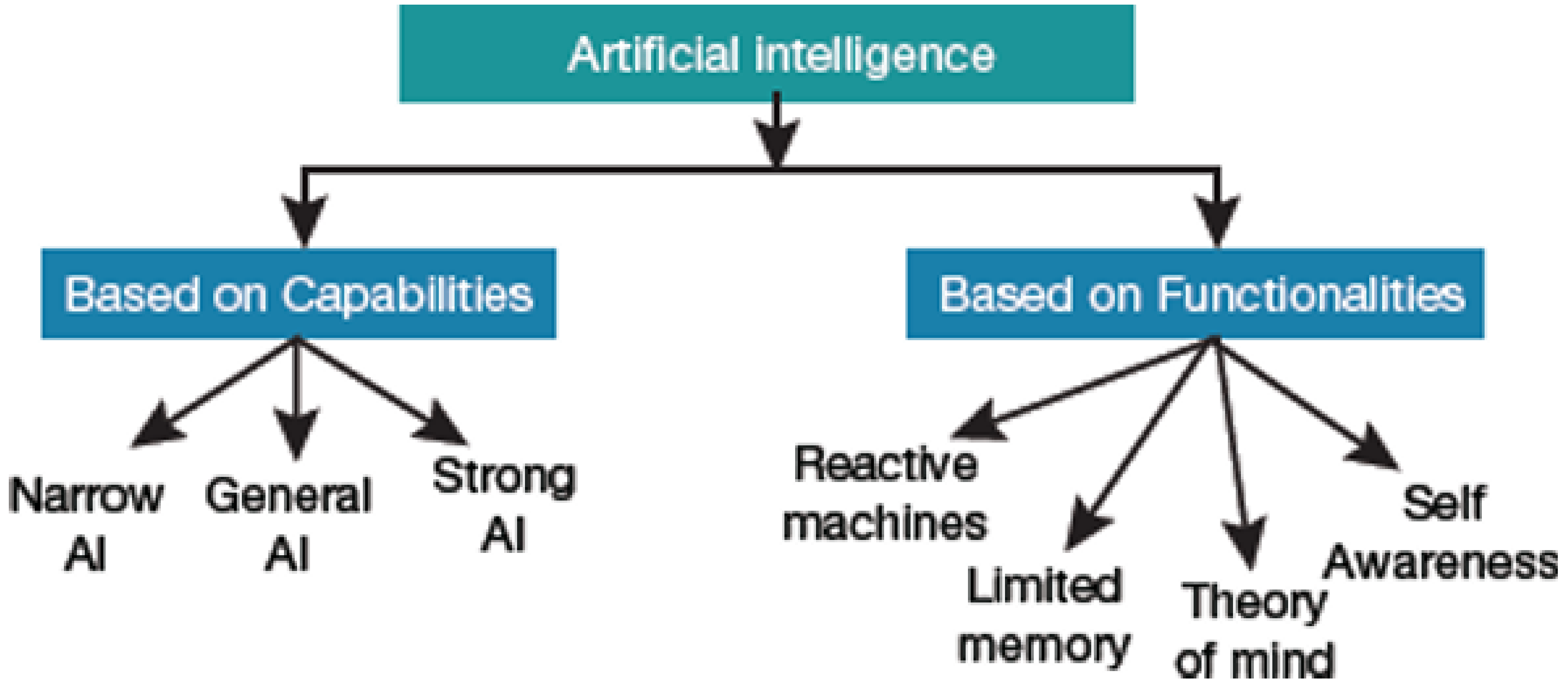
- **Weak AI (Narrow AI)**
    - Weak AI, also known as narrow AI, is designed to do one specific task.
    - Siri and Alexa are examples.
- **Strong AI (Artificial General Intelligence)**
    - Strong AI, also called Artificial General Intelligence (AGI) or Superintelligence (ASI), tries to mimic human thinking.
    - Can perform tasks it hasn't been specifically trained for.
    - Requires abilities like visual perception, speech recognition, decision-making, and language translation.
- Reactive Machines
    - Reactive Machines are the simplest type of AI that react to situations based on immediate input, but they have no memory or ability to learn from past experiences.
    - Examples: IBM's Deep Blue (chess-playing computer) is a reactive machine. It makes decisions based on the current state of the game but doesn't remember past games.



Types of AI

AI can be categorized into:

- **Based on Capabilities**
  - **Weak AI (Narrow AI)**
    - Weak AI, also known as narrow AI, is designed to do one specific task.
  - **Strong AI**
    - Strong AI, also called Artificial General Intelligence (AGI) or Superintelligence (ASI), tries to mimic human thinking



Module 1: Introduction to Artificial Intelligence

Types of AI

AI can be categorized into:

- **Based on Functionalities**
  - **Reactive Machines**
    - Reactive Machines are the simplest type of AI that react to situations based on immediate input, but they have no memory or ability to learn from past experiences.
  - **Limited Memory**
    - Limited memory AI systems can remember data for a short time and use it to make decisions, but they don't keep data permanently.
  - **Theory of Mind**
    - The Theory of Mind in AI aims to create machines that can understand thoughts, emotions, and memories—just like humans.
  - **Self-Awareness**
    - Self-awareness in AI means machines that have a human-level consciousness—they can understand their own existence and feelings.

Module 1: Introduction to Artificial Intelligence

Types of AI

1. **Weak AI (Narrow AI)**
   - Weak AI, also known as narrow AI, is designed to do one specific task.
   - **Examples:**
     - Siri and Alexa are examples. When you tell Alexa to play a song, it does so because it's trained to understand that specific command.
     - Other examples include weather forecasting, predicting stock prices, and Google search.
   - **How it works:** These systems are great at doing one thing really well, but they don't work outside their specific task. For example, Alexa can't drive a car; it's just built for voice commands.
   - **Why it's important:** Weak AI has helped make many tasks easier and more efficient, and it is the most common type of AI in use today.

Types of AI

2. Strong AI (Artificial General Intelligence - AGI)

- Strong AI, also called Artificial General Intelligence (AGI) or Superintelligence (ASI), tries to mimic human thinking.
- Can perform tasks it hasn't been specifically trained for.
- Requires abilities like visual perception, speech recognition, decision-making, and language translation.
- **Future potential:** Experts believe that Strong AI might one day surpass human intelligence, but it's not expected to happen anytime soon.
- **Concerns:** While some fear that Strong AI could be dangerous, experts say we don't need to worry about it in the near future, as it's still far from becoming a reality.

Types of AI

3. Reactive Machines

- **Reactive Machines** are the simplest type of AI that react to situations based on immediate input, but they have no memory or ability to learn from past experiences.
  - **Examples:** IBM's Deep Blue (chess-playing computer) is a reactive machine. It makes decisions based on the current state of the game but doesn't remember past games.
-

Module 1: Introduction to Artificial Intelligence

Types of AI

4. Limited Memory

- What it is: Limited memory AI systems can remember data for a short time and use it to make decisions, but they don't keep data permanently.
    - **Examples:**
        - Autonomous vehicles use limited memory to track information like speed of nearby cars, distance between cars, and speed limits to navigate safely.
        - AlphaGo, the AI that defeated the world champion in the game Go, also used limited memory to play and improve during the game.
    - These systems learn and improve continuously by analyzing new data and adjusting based on feedback.
    - **Key Models:**
        - **Reinforcement Learning:** AI learns by trial and error, improving over time.
        - **Long Short-Term Memory (LSTM):** AI uses past data to predict the next step, but it focuses more on recent data.
        - **Evolutionary GANs (E-GAN):** The AI evolves over time, using data and feedback to make better decisions and predict outcomes.

Module 1: Introduction to Artificial Intelligence

Types of AI

5 Theory of Mind

- The Theory of Mind in AI aims to create machines that can understand thoughts, emotions, and memories—just like humans.
- **How it works:** AI would need to understand feelings and emotions that influence decisions. These machines would make choices by considering both reason and emotional context.
- **Current Status:** This is still theoretical, meaning it's an idea for the future, but it could become a reality soon.

Types of AI

6 Self-Awareness

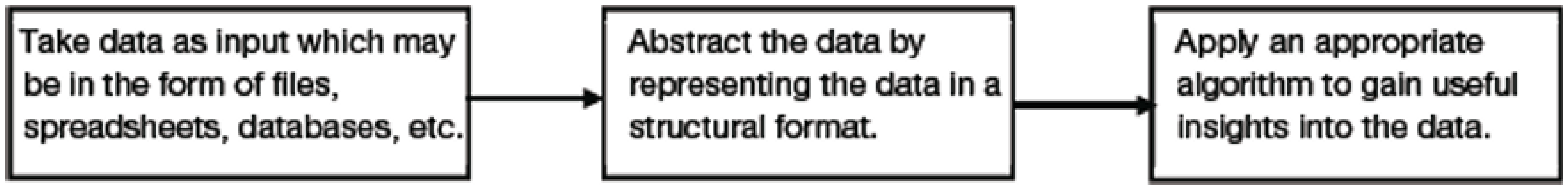
- **What it is:** Self-awareness in AI means machines that have a human-level consciousness—they can understand their own existence and feelings.
- **How it works:** These machines would not only understand what someone says but also how they feel based on the way they communicate. They could learn and adapt their responses to the emotions of others.
- **Current Status:** Self-awareness in AI doesn't exist yet, but it might happen in the future.

Is AI the Same as Augmented Intelligence and Cognitive Computing?

AI vs Augmented Intelligence:

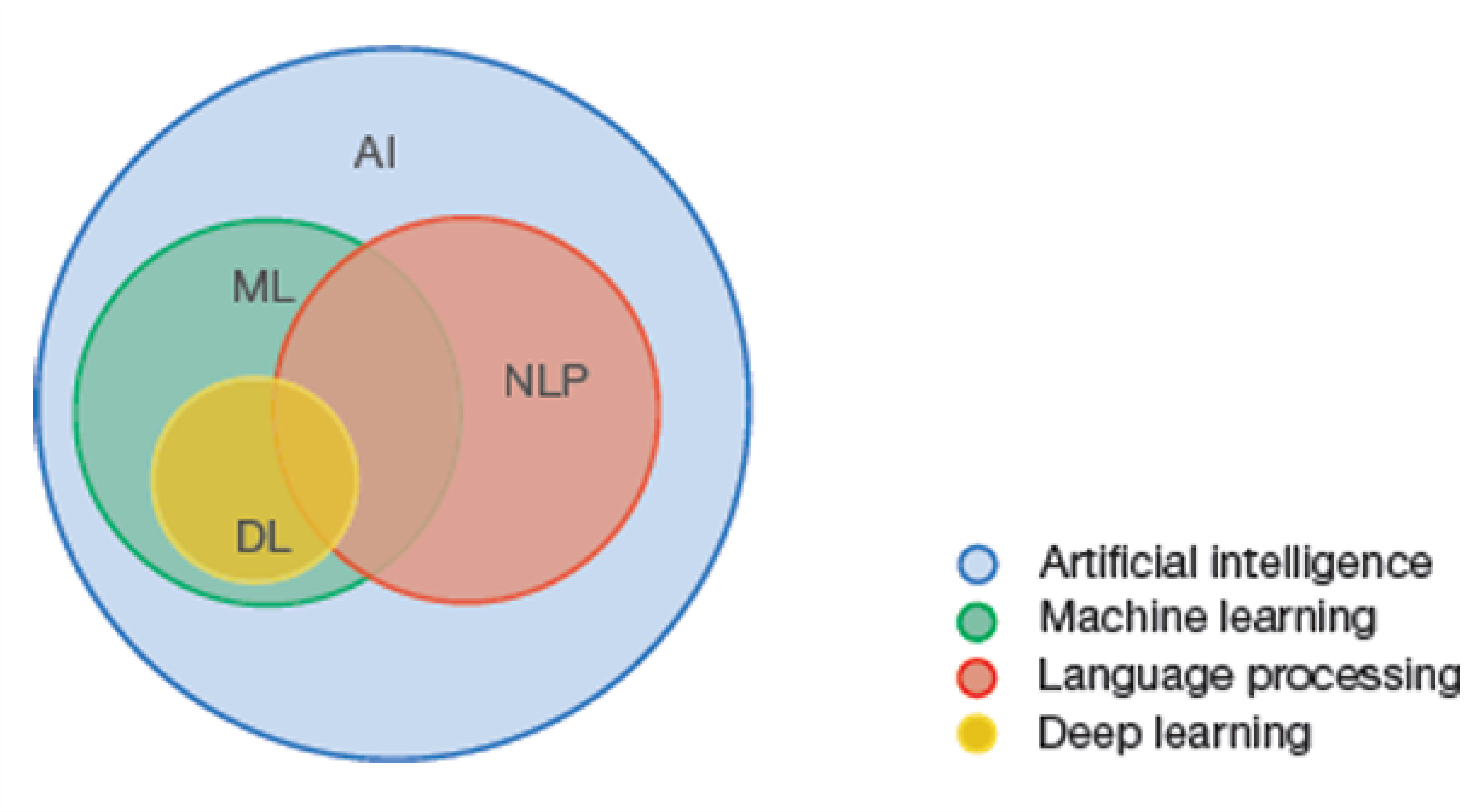
- Some people think AI and augmented intelligence are the same, but they are different.
- Augmented intelligence is a type of weak AI that assists humans to improve tasks or decisions.
  - Example: Automatically highlighting important information in a report.
- True AI / Strong AI / AGI is future AI that aims to surpass human intelligence, capable of performing tasks that humans can do, like reasoning and decision-making.
- **AI in Machines:** AI makes machines simulate human intelligence by learning, sensing, processing, and reacting to information.
- **Cognitive Computing:** This refers to systems or products that mimic human thought processes to enhance decision-making.
  -

Machine Learning (ML)

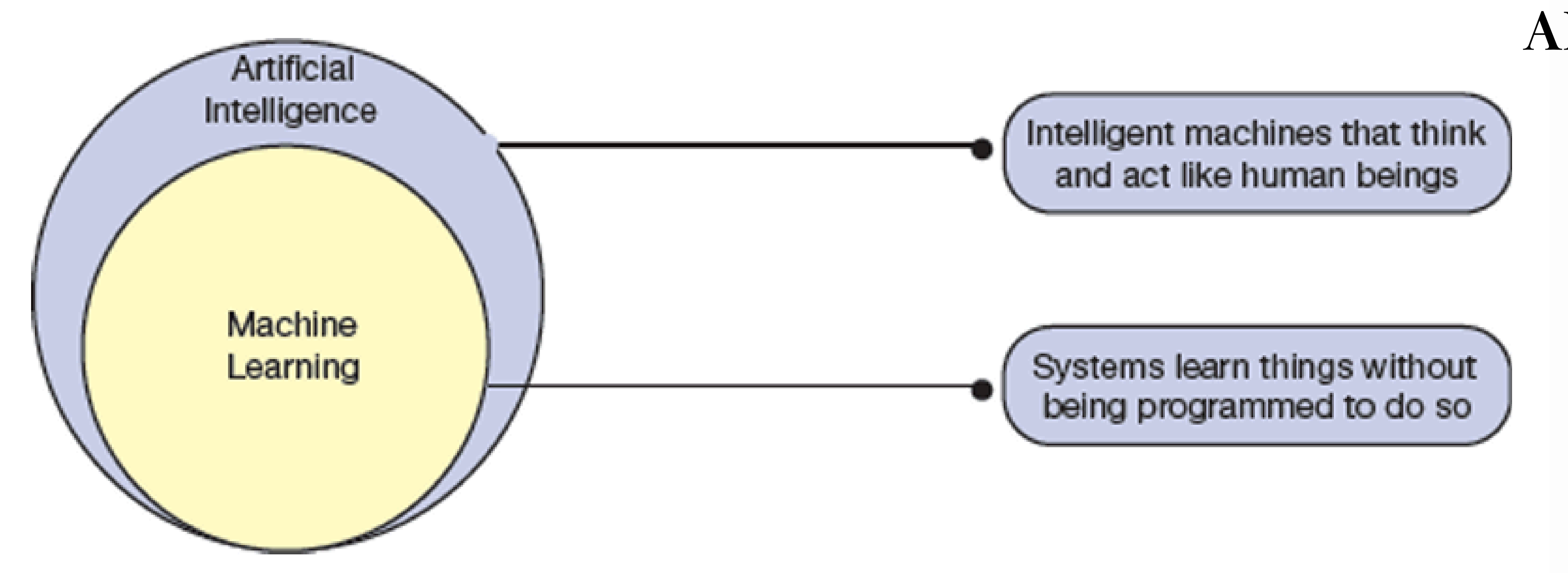
- Machine Learning is a branch of AI that teaches machines to learn from data and make decisions without being explicitly programmed.
- How it works:
    a. Finding patterns: ML algorithms analyze data to identify patterns.
    b. Learning from experience: Machines improve automatically by learning from their past output.
    c. Self-correction: If the machine makes a mistake, it adjusts and learns to improve accuracy over time.
- Real-life examples:
    - Number series problem: Finding the missing number in 10, 20, 30, 40 → 50. Machines learn patterns just like humans do.
    - 

Module 1: Introduction to Artificial Intelligence

Relationship between Artificial Intelligence, Machine Learning, Deep Learning and Natural Language Processing



HOW IS AI RELATED TO MACHINE LEARNING?



AI vs ML:

- AI is the larger goal (the superset) that aims to make machines intelligent.
- Machine Learning is a subset of AI, used specifically for learning from data to make decisions.
- In simple terms, ML helps achieve AI by making machines learn and adapt.

Example:

- If you want a robot that can see, talk, walk, and learn, you would use AI because it requires many different technologies.
- Machine Learning would only be used to help the robot learn from its environment or past experiences.

Module 1: Introduction to Artificial Intelligence

Traditional Programming vs Machine Learning

Traditional Programming:

- Manual Coding: In traditional programming, the programmer manually writes code that accepts input data and returns output based on pre-defined rules.
- Languages Used: It uses procedural programming languages like C, C++, Java, Python, etc., where the logic and rules are explicitly coded by the programmer.
- Algorithm-Dependent: The program is created based on specific algorithms chosen by the programmer, who also analyzes their performance to pick the best one for the task.
- Static Rules: The rules and logic of the program are fixed and cannot change unless modified by the programmer.

Machine Learning Programming:

- **Data-Driven Approach:** Machine learning programming learns automatically from data
- **Predictive Modeling:** For example, if we input customer data and transactions, machine learning can create a predictive model to forecast
- **Automated Learning:** Adapt and improve over time as more data is processed.
- **Embedded Analytics:** Machine learning introduces embedded analytics, like natural language processing, automatic anomaly detection, and recommendation systems, to make intelligent predictions

Traditional Programming vs Machine Learning

Key Differences:

- **Traditional Programming:** It uses fixed logic and pre-written rules defined by the programmer.
- **Machine Learning:** It learns from data, automatically creates its own rules, and improves over time based on experience.

Example:

- Traditional Programming: To filter images manually, you would have to write code comparing each pixel value in the image. This approach can be slow and inaccurate.
- Machine Learning: You simply provide photos of a person, and the model learns to recognize that person based on patterns in the images, making the task much easier and more efficient.

Machine Learning and Deep Learning

Artificial Intelligence (AI):

- AI is the broad concept of creating machines or systems that mimic human intelligence. It involves a variety of algorithms and approaches, one of which is machine learning.

Machine Learning (ML):

- ML is a specific subset of AI that involves teaching machines to learn from data.
- ML techniques can be divided into:
  - Supervised Learning: Uses labeled data (data with known outcomes) to train the model.
  - Unsupervised Learning: Uses unlabeled data (data without predefined outcomes) to discover patterns or structure in the data.

Deep Learning (DL):

- Deep learning is a subset of machine learning that uses neural networks with multiple layers (known as deep neural networks) to analyze and process large amounts of data. These networks are inspired by the way the human brain processes information.
- Deep learning allows machines to learn in much more complex ways, making connections between layers of data to improve decision-making.
- It is particularly powerful for tasks like image recognition, speech processing, and other complex tasks.

How Does AI Work?

- **Study Large Data:** AI analyzes large amounts of data to find patterns.
- **Make Predictions:** AI makes predictions based on the data it has studied.
- **Autonomous Decision Making:** AI can make decisions on its own, learning from past data and experiences.
- **Adapt and Improve:** AI systems continuously adapt and learn from new data to improve over time.
- **React to Problems:** AI systems perceive problems and respond accordingly, using the patterns they have learned.
- **Data-Driven:** AI relies on large amounts of data to make decisions, with cheap data storage and fast processors making it easier to use.
- **Make Accurate Predictions:** AI can make accurate predictions based on past data and experience.
- **Growing and Evolving:** AI is being applied in many different areas and its scope is continuously expanding.

Module 1: Introduction to Artificial Intelligence

Chapter 3: Artificially Intelligent Machine

3.1 Defining Intelligence,

3.2 Components of Intelligence,

3.3 Differences Between Human and Machine Intelligence,

3.4 Agent and Environment,

3.5 Search,

3.6 Uninformed Search Algorithms,

3.7 Informed Search Algorithms:

Module 1: Introduction to Artificial Intelligence

3.1 Defining Intelligence

- **Linguistic Intelligence:**
  - The ability to speak and understand language (e.g., narrators, orators).
- **Musical Intelligence:**
  - The ability to create and understand music, like recognizing rhythm and pitch (e.g., musicians, singers).
- **Logical-Mathematical Intelligence:**
  - The ability to understand abstract concepts and use logic (e.g., mathematicians, scientists).
- **Spatial Intelligence:**
  - The ability to visualize and manipulate images in your mind (e.g., map readers, astronauts).
- **Bodily-Kinesthetic Intelligence:**
  - The ability to use your body to solve problems or manipulate objects (e.g., dancers, athletes).
- **Intrapersonal Intelligence:**
  - The ability to understand your own feelings and motivations (e.g., spiritual leaders, philosophers).
- **Interpersonal Intelligence:**
  - The ability to understand other people's feelings and intentions (e.g., mass communicators, interviewers).
- **Artificial Intelligence (AI):** A system or machine is said to be artificially intelligent if it can exhibit one or more of these types of intelligence.

3.2 Components of Intelligence

- Reasoning
- Learning
- Problem solving
- Perception
- Linguistic intelligence

3.2 Components of Intelligence - 1. Reasoning

Reasoning is the process used to make decisions and predictions. It involves analyzing information and drawing conclusions based on evidence. There are two main types of reasoning:

- **1. Inductive Reasoning**:
  - What it is: Making generalizations based on specific observations or examples.
  - How it works: Starts with specific facts or observations and moves to a general conclusion.
  - Example: If you see 10 white swans, you might conclude that all swans are white.
- 2. Deductive Reasoning:
  - What it is: Drawing a specific conclusion based on general principles or facts.
  - How it works: Starts with a general statement or premise, and moves to a specific conclusion.
  - Example: All swans are birds. A swan is a bird. Therefore, the swan is a bird.

3.2 Components of Intelligence - 2. Learning

Learning is the process of gaining knowledge or skills by studying, practicing, or experiencing something. It helps humans, animals, and even AI systems to improve their understanding of different subjects.

There are different types of learning:

1. Auditory Learning:
2. Episodic Learning:
3. Motor Learning:
4. Observational Learning:
5. Perceptual Learning:
6. Relational Learning:
7. Spatial Learning:
8. Stimulus-Response Learning:

3.2 Components of Intelligence - 2. Learning

There are different types of learning:

1. Auditory Learning:
   - Learning by hearing and listening.
   - Example: Listening to recorded lectures to understand a concept.
2. Episodic Learning:
   - Learning by remembering events or experiences in a specific order.
   - Example: Recalling what happened in a sequence, like remembering steps in a recipe.
3. **Motor Learning:**
   - Learning through physical movement of muscles.
   - Example: Learning how to pick up objects correctly.
4. **Observational Learning:**
   - Learning by watching and imitating others.
   - Example: Children learn by copying their parents' actions.

3.2 Components of Intelligence - 2. Learning

5. **Perceptual Learning:**
   - Learning by recognizing things that have been seen before.
   - Example: Identifying objects and situations based on prior experiences.
6. **Relational Learning:**
   - Learning by recognizing patterns in relationships between things.
   - Example: Adjusting the amount of spices in a dish after remembering how much was used last time.
7. **Spatial Learning:**
   - Learning through visual stimuli like images, maps, and colors.
   - Example: Creating a mental map of a route before actually driving.
8. **Stimulus-Response Learning:**
   - Learning by reacting to a specific stimulus.
   - Example: Shouting when touching a hot pan, because it causes pain.

3.2 Components of Intelligence - 3. Problem Solving

Problem solving is the process of finding a solution to a challenge or issue. It involves:

- Identifying the problem: Understanding the situation and recognizing the obstacles (either known or unknown).
- Making decisions: Choosing the best approach or method to overcome the obstacles and reach the goal.

3.2 Components of Intelligence - 4. Perception

Perception is the process of:

- Acquiring information through the senses (like sight, hearing, etc.).
- Interpreting that information to understand what's happening around us.
- Selecting important details and organizing them to form a clear picture.
- Humans use sensory organs (like eyes, ears) to perceive the world.
- AI systems use sensors (like cameras, microphones) to gather data and understand their environment.

3.2 Components of Intelligence - 5. Linguistic Intelligence

It is used in in interpersonal communication and defines one's ability to use, comprehend, speak and write the verbal and written language

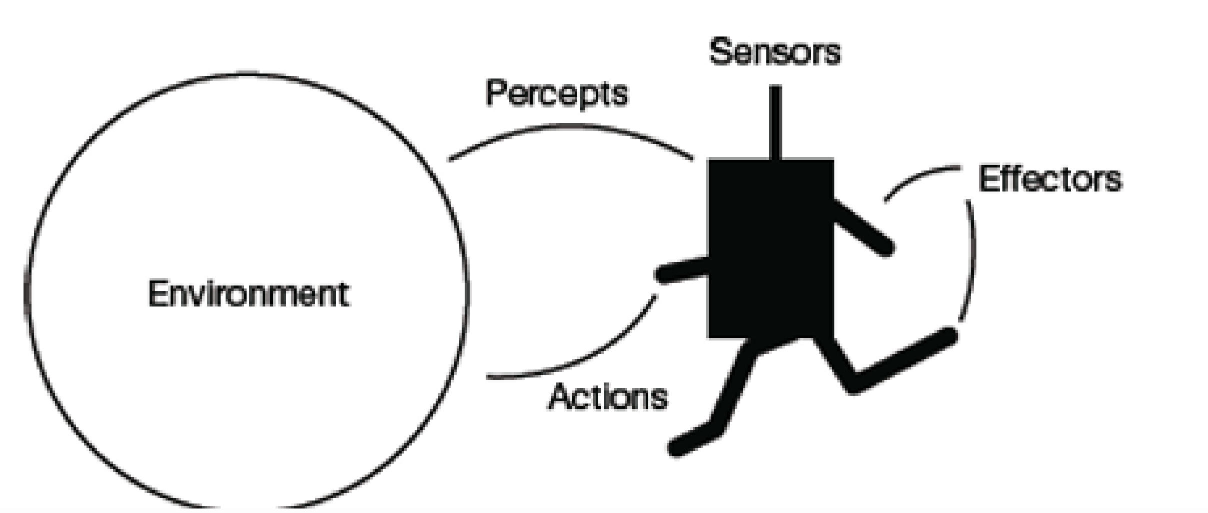
3.3 Differences Between Human and Machine Intelligence

Aspect	Human Intelligence	Machine Intelligence
Perception	Perceives through patterns	Perceives by analyzing data with rules
Memory and Recall	Recalls information by patterns	Uses search algorithms to find information
Handling Missing Information	Can deduce missing or distorted information accurately	Struggles with incomplete data, less accurate

3.4 Agent and Environment

AI agents act in their environment, which may include other agents.

- They perceive their environment using **sensors.**
- They act upon the environment using **effectors.**



Types of Agents in an AI System:

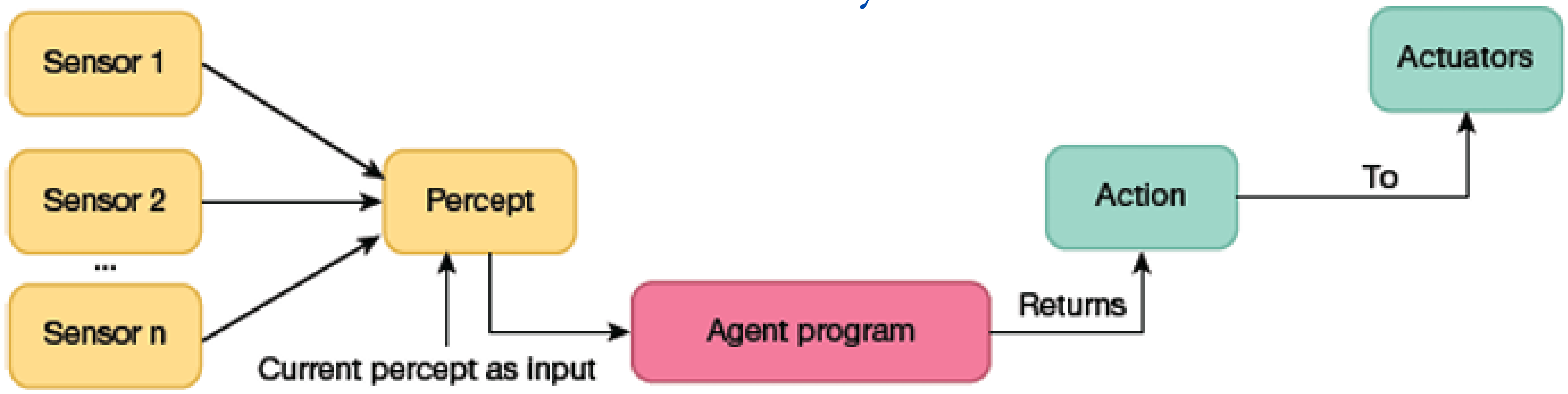
1. **Human Agent:**
   - Sensors: Sensory organs like eyes, ears, nose, skin, etc.
   - Effectors: Hands, legs, mouth for taking action.
2. **Robotic Agent:**
   - Sensors: Cameras, infrared range finders.
   - Effectors: Motors, actuators to perform actions.
3. **Software Agent:**
   - Sensors: Uses bit strings as its programs.
   - Effectors: Executes programmed actions based on those bit strings.

3.4 Agent and Environment - 3.4.1 Key Terminology

- **Performance Measure of Agent:**
  - It helps determine how successful an agent is based on its actions.
- **Behaviour of Agent:**
  - The action performed by an agent after receiving a percept (input).
- **Percept:**
  - Perceptual input received by an agent at a specific moment in time.
- **Percept Sequence:**
  - A list of all percepts an agent has received up until now.
- **Agent Function:**
  - A map that connects the percept sequence to an action performed by the agent.
  -

Module 1: Introduction to Artificial Intelligence

3.4 Agent and Environment - 3.4.2 Rationality



- Rationality is the ability to make responsible and sensible decisions.
- A rational agent makes decisions that maximize its performance based on:
    a. Performance measure (how successful the agent is).
    b. Percept sequence (the inputs it has received).
    c. Prior knowledge (what the agent already knows about the environment).
    d. Possible actions (what the agent can do).
- A rational agent always performs the right action to maximize its performance.
- Problem Solved by Agent (PEAS):
- Performance measure, Environment, Actuators, and Sensors are used to define a problem that an agent will solve.

3.4 Agent and Environment - 3.4.4 Types of Agents

1. Simple Reflex Agents
2. Model-Based Reflex Agents
3. Goal-Based Agents
4. Utility-Based Agents
5. Learning Agent

3.4 Agent and Environment - 3.4.4 Types of Agents

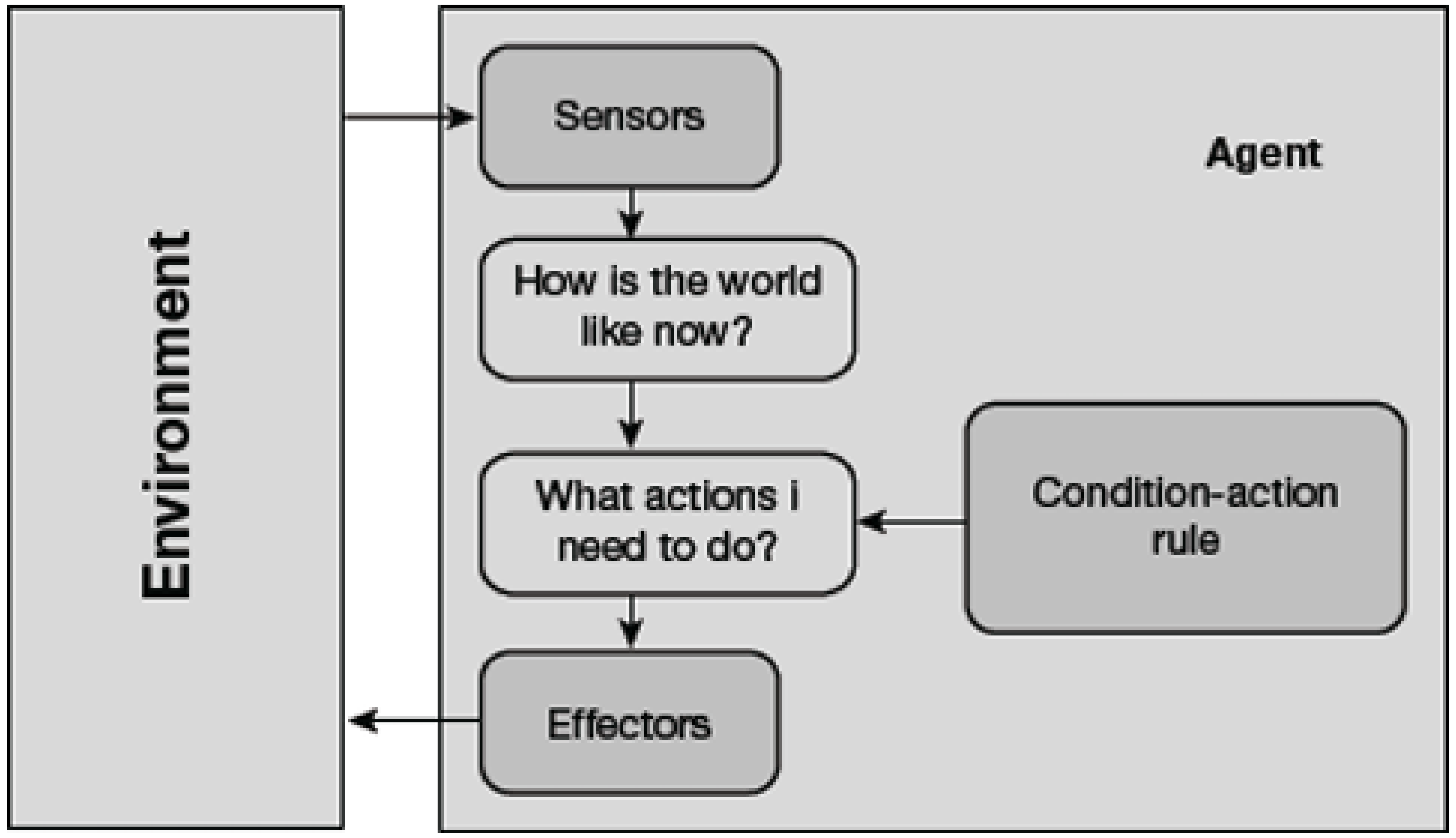
1. Simple Reflex Agents

Simple reflex agents choose actions based only on the current percept (the data they receive at a specific moment).

- They are rational only if they make the correct decision based on the current percept.
- **Working:**
  - They use a condition-action rule that maps a state (condition) to an action.
  - If the condition is true, the agent performs the action; otherwise, it does nothing.
- **Limitations:**
  - They require the environment to be fully observable (the agent must have access to complete information).
  - If the environment is partially observable, the agent might get stuck in infinite loops.
  - In such cases, the agent can only escape the loop if it randomizes its actions.
- **Other Issues:**
  - Simple reflex agents have very limited intelligence.
  - They don't know anything about states other than the current one.
  - If the environment changes, the rules they follow might need to be updated.

Module 1: Introduction to Artificial Intelligence

3.4 Agent and Environment - 3.4.4 Types of Agents

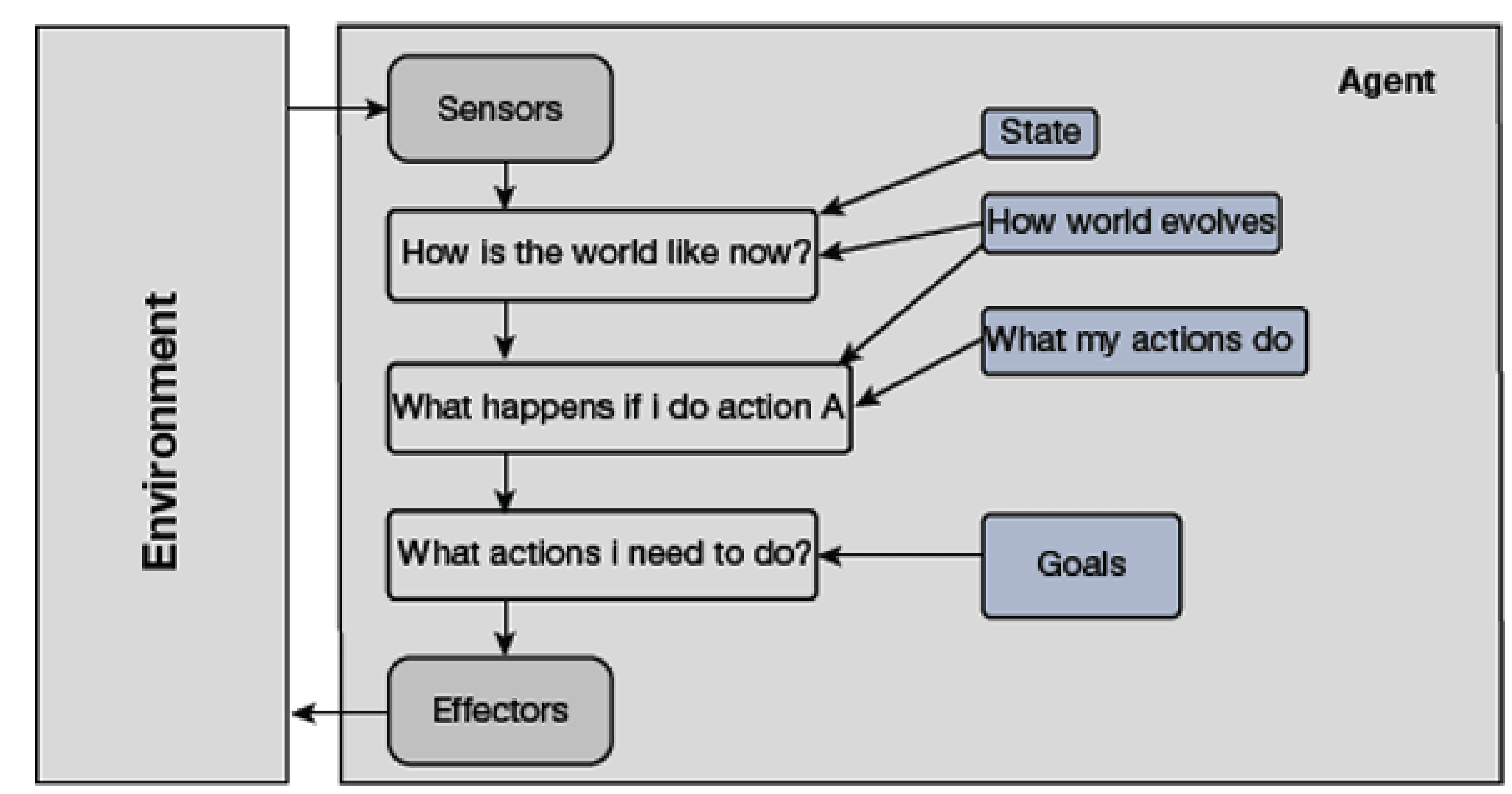


2. Model-Based Reflex Agents

- Model-based reflex agents use a model of the world to choose their actions, which requires them to maintain an internal state.
- **Internal State:**
  - The internal state represents aspects of the current situation that are not directly observable but can be inferred based on the history of percepts.
- **Working**
  a. The agent updates its internal state by understanding two things:
  b. How the world evolves (what happens over time in the environment).
  c. How the agent's actions affect the world (the consequences of the agent's actions).

Module 1: Introduction to Artificial Intelligence

3.4 Agent and Environment - 3.4.4 Types of Agents

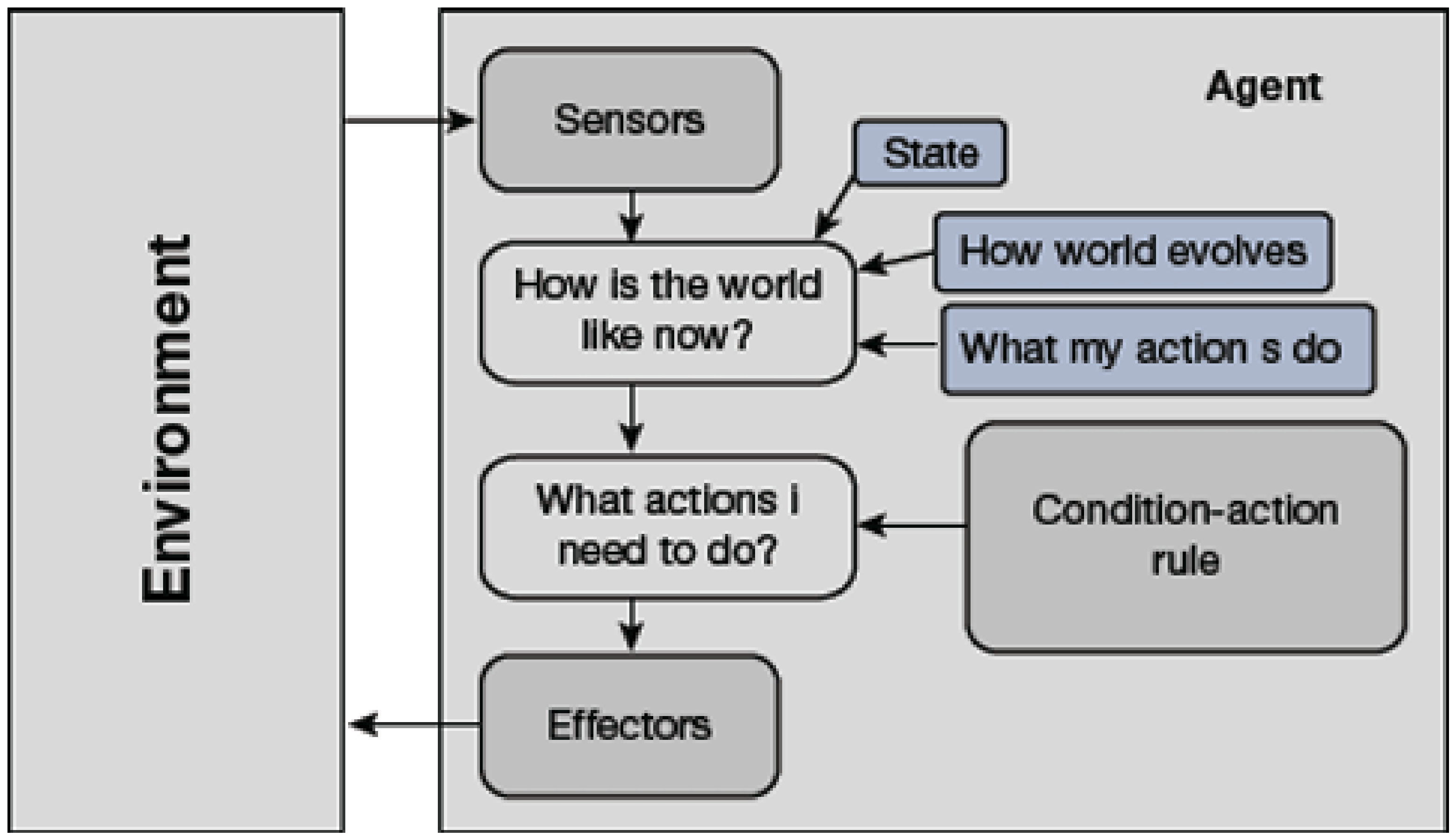


3. Goal-Based Agents

- A description of desirable situations or outcomes.
- **Function:** Goal-based agents choose their actions to achieve specific goals.
- **Flexibility:** Provides more flexibility than reflex agents as the decision-making knowledge is explicitly modeled.
- **Modifications:** Allows modifications to adapt to changing conditions.

Module 1: Introduction to Artificial Intelligence

3.4 Agent and Environment - 3.4.4 Types of Agents

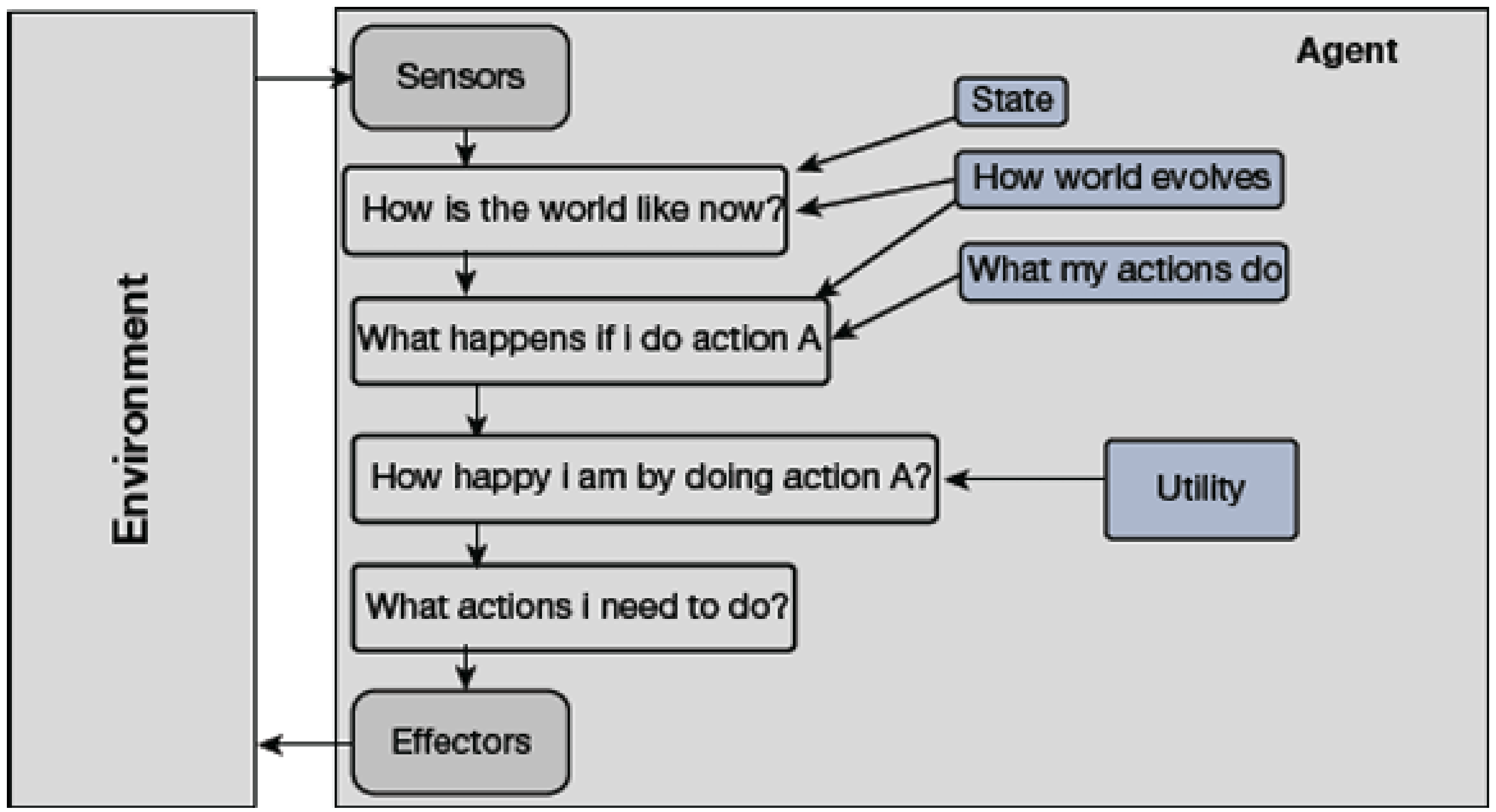


4. Utility-Based Agents

- **Purpose:** Used when goals are conflicting or difficult to achieve.
- **Function:** These agents choose actions based on a preference (utility) for each state.
- **Goal:** They help prioritize goals by choosing actions that lead to the most preferred outcome.
- **Flexibility:** Provides a way to handle multiple goals by considering their relative importance.

Module 1: Introduction to Artificial Intelligence

3.4 Agent and Environment - 3.4.4 Types of Agents



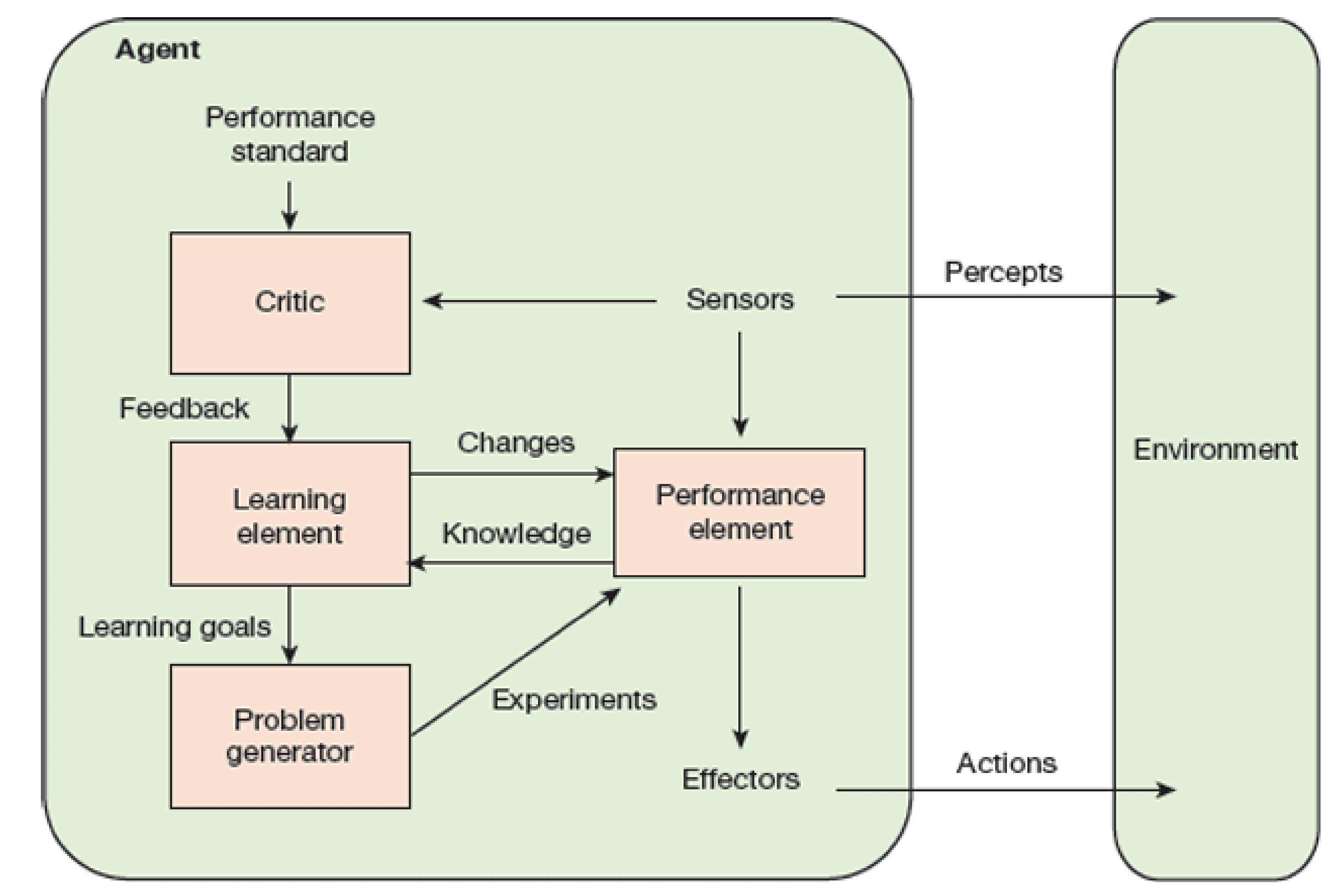
5. Learning Agent

- Purpose: A learning agent learns from past experiences and adapts over time.
- Starting Point: It begins with basic knowledge and gradually improves by learning from its environment.

Four Main Components:

1. Learning Element:
   - Responsible for making improvements by learning from the environment.
2. Critic:
   - Provides feedback to the learning element, evaluating how well the agent is performing against a set performance standard.
3. Performance Element:
   - Selects the external action to be taken based on the current situation.
4. Problem Generator:
   - Suggests actions that lead to new experiences for the agent, promoting further learning.

3.4 Agent and Environment - 3.4.5 The Nature of Environments



Artificial Intelligence Systems:

- Some AI programs are confined to limited environments like keyboard input, databases, and file systems.
- Others, like software robots (softbots), operate in unlimited domains with complex environments.

Turing Test

- Purpose: The Turing Test is used to determine if a machine can exhibit intelligent behavior.
- Test Setup:
  a. Two humans and a software agent (machine) participate.
  b. One human is the tester, unaware of which participant is the machine.
  c. The tester sends typed questions to both humans and the software agent.

3.4 Agent and Environment - 3.4.6 Types of Environments

1. **Discrete/Continuous:**
   a. Discrete: Limited, well-defined states (e.g., chess).
   b. Continuous: No limitations on percepts or actions (e.g., self-driving car).
2. **Known vs Unknown:**
   a. Known: Agent knows results for all actions.
   b. Unknown: Agent must learn how to act (e.g., reinforcement learning).
3. **Observable/Partially Observable:**
   a. Observable: Agent can perceive the complete state (e.g., chess).
   b. Partially Observable: Agent cannot perceive everything (e.g., Kriegspiel chess).
4. **Static/Dynamic:**
   a. Static: Environment does not change while acting (e.g., crossword puzzle).
   b. Dynamic: Environment changes during action (e.g., self-driving car).
   c. Semi-dynamic: Environment doesn't change, but agent's performance can change.

3.4 Agent and Environment - 3.4.6 Types of Environments

5. Single Agent/Multiple Agents:

- Single Agent: One agent in the environment (e.g., vacuum cleaner).
- Multiple Agents: More than one agent, can be competitive or cooperative (e.g., chess, taxi driving).

6. Accessible/Inaccessible:

- Accessible: Agent has full access to environment information (e.g., empty room).
- Inaccessible: Agent cannot get complete information (e.g., global events).

7. Deterministic/Non-deterministic:

- Deterministic: Next state can be determined from current state (e.g., chess).
- Non-deterministic: Uncertainty about outcomes (e.g., ludo, dice roll).

8. Episodic/Non-episodic:

- Episodic: Each episode is independent (e.g., simple tasks).
- Non-episodic: Current actions affect future actions (e.g., long-term decision making).

3.5 Search

AI agents use search algorithms to solve tasks and make decisions. For example, single-player games like Sudoku and tile games use search algorithms to find optimal moves or positions.

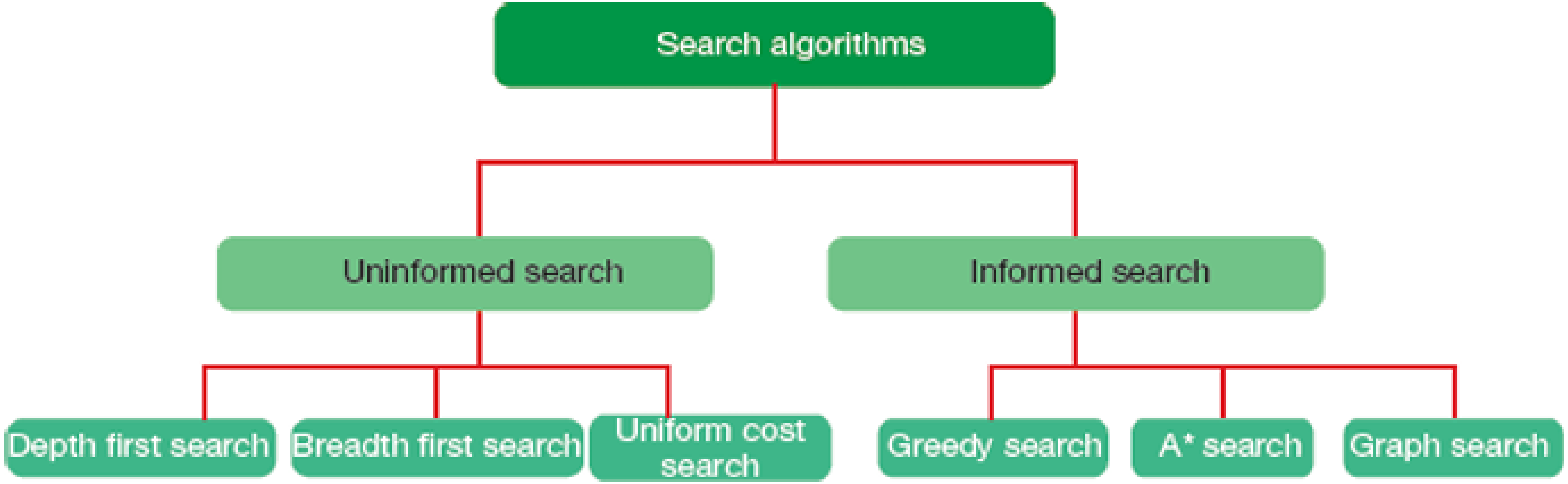
Components of a Search Problem

1. **State Space:** The set of all possible states the agent can reach.
2. **Start State:** The initial state where the search begins.
3. **Goal Test:** A function that checks if the current state is the goal state.
4. **Solution:** A sequence of actions (plan) that transforms the start state to the goal state, achieved using search algorithms.

Module 1: Introduction to Artificial Intelligence

3.5 Search

3.5.1 Types of Search Algorithms



3.5 Search

3.5.2 Properties of Search Algorithms

- **Completeness:** A search algorithm is complete if it guarantees at least one solution for a given input.
- **Optimality:** A search algorithm is optimal if it provides the best solution with the lowest path cost.
- **Time and Space Complexity:**
  - **Time Complexity:** The amount of time an algorithm takes to complete a task.
  - **Space Complexity:** The amount of memory required for the search process.
  - A good search algorithm should use less time and less memory.

3.6 Uninformed Search Algorithms

- Uninformed search (or blind search) algorithms have no extra information about the goal state other than what is provided in the problem definition. The algorithm blindly explores the search space without considering how close it is to the goal.
- **Key Concepts:**
  a. **Problem Graph: R**epresents the problem, from the start node (S) to the goal node (G).
  b. **Strategy:** The path taken in the search to reach the goal.
  c. **Fringe:** A data structure that stores all possible states (nodes) that can be reached from the current state.
  d. **Tree:** The path representation that the algorithm follows while searching for the goal node.
  e. **Solution Plan:** The sequence of nodes (states) from start node (S) to goal node (G).
  f. **Path/Step Cost:** Integer values that represent the cost to move from one node to another.

3.6 Uninformed Search Algorithms

- Uninformed search (or blind search) algorithms have no extra information about the goal state other than what is provided in the problem definition. The algorithm blindly explores the search space without considering how close it is to the goal.
- **Key Concepts:**
  a. **Problem Graph: R**epresents the problem, from the start node (S) to the goal node (G).
  b. **Strategy:** The path taken in the search to reach the goal.
  c. **Fringe:** A data structure that stores all possible states (nodes) that can be reached from the current state.
  d. **Tree:** The path representation that the algorithm follows while searching for the goal node.
  e. **Solution Plan:** The sequence of nodes (states) from start node (S) to goal node (G).
  f. **Path/Step Cost:** Integer values that represent the cost to move from one node to another.

3.6.1 Depth First Search (DFS)
3.6.2 Depth-Limited Search Algorithm (DLS)
3.6.3 Breadth First Search (BFS)
3.6.4 Uniform Cost Search (UCS)
3.6.5 Iterative Deepening Depth-First Search (IDDFS)
3.6.6 Bidirectional Search

3.6 Uninformed Search Algorithms - 3.6.1 Depth First Search (DFS)

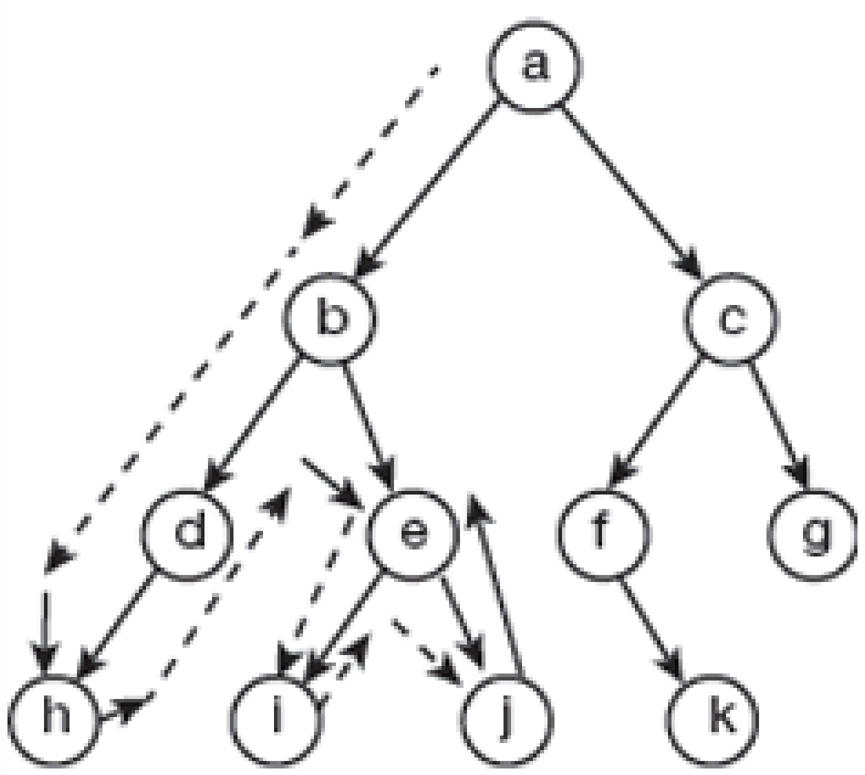
Depth First Search (DFS) is a simple search algorithm used to explore a tree or graph by starting from the root node and exploring as far as possible along each branch before backtracking.

Steps of DFS:

1. **Start from the root node:** Begin searching from the root node (node A).
2. **Explore each branch:** Move from node A to its child node (B), then to the next child (D), and continue exploring until you reach the leaf node (the last node of that branch).
3. **Backtrack:** If the key you're looking for isn't found at the leaf node, backtrack to the last node with unexplored branches and explore them.
4. **Repeat the process:** Continue exploring each branch by backtracking and then moving to the next unexplored branch, until the entire tree is searched or the goal is found.

Module 1: Introduction to Artificial Intelligence

3.6 Uninformed Search Algorithms - 3.6.1 Depth First Search (DFS)



- Start at node A.
- Explore node B, then node D, then node H (leaf node).
- Since H is a leaf and the key isn't found, backtrack to node B and explore node E, then node I (leaf node).
- Backtrack again to explore node J and other branches of node B.
- Once all branches of node B are explored, move to node C, then node F, node K, and finally node G.

Key Points:

- Backtracking is a key feature of DFS, meaning it explores as far as possible along each branch before going back to previous nodes to try other paths.
- DFS uses a stack (Last-In-First-Out or LIFO structure) to keep track of nodes to visit next.

3.6 Uninformed Search Algorithms - 3.6.1 Depth First Search (DFS)

Advantages of Depth First Search (DFS)

1. **Less Memory Usage:** DFS stores only the nodes along the path from the root node to the current node, requiring less memory.
2. **Faster to Reach Goal:** It often takes less time to find a goal compared to Breadth First Search (BFS), especially when the solution is deep in the tree.

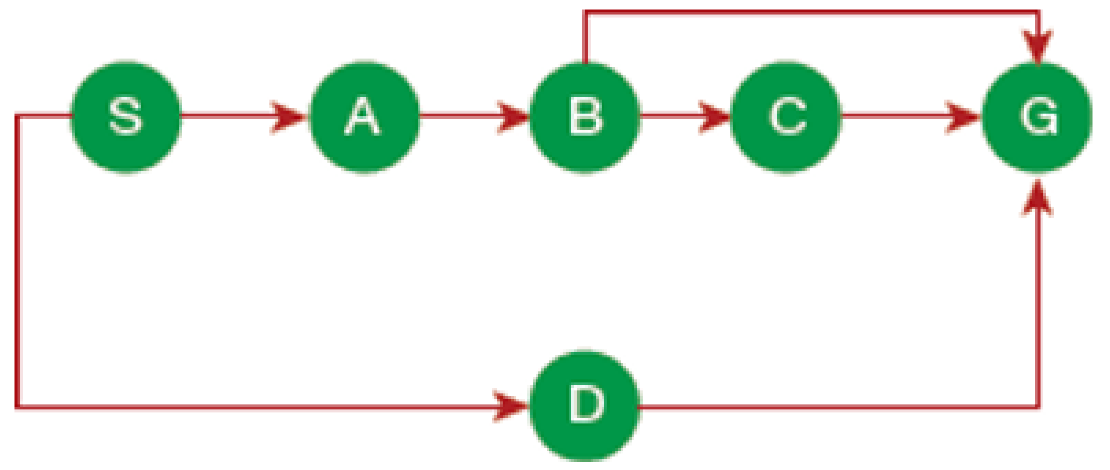
Disadvantages of Depth First Search (DFS)

1. **Recurring States:** Sometimes, many states repeat. In such cases, there's no guarantee of finding the solution.
2. **Infinite Loops:** DFS may get stuck in an infinite loop when it keeps going deeper. This can be avoided by setting an appropriate cut-off depth, but:
   - Too small a cut-off may make the algorithm fail.
   - Too large a cut-off increases execution time.
3. **Complexity:** The algorithm's complexity depends on the number of paths it needs to explore.
4. **Duplicate Nodes:** DFS cannot check for duplicate nodes, potentially leading to inefficiency in the search.

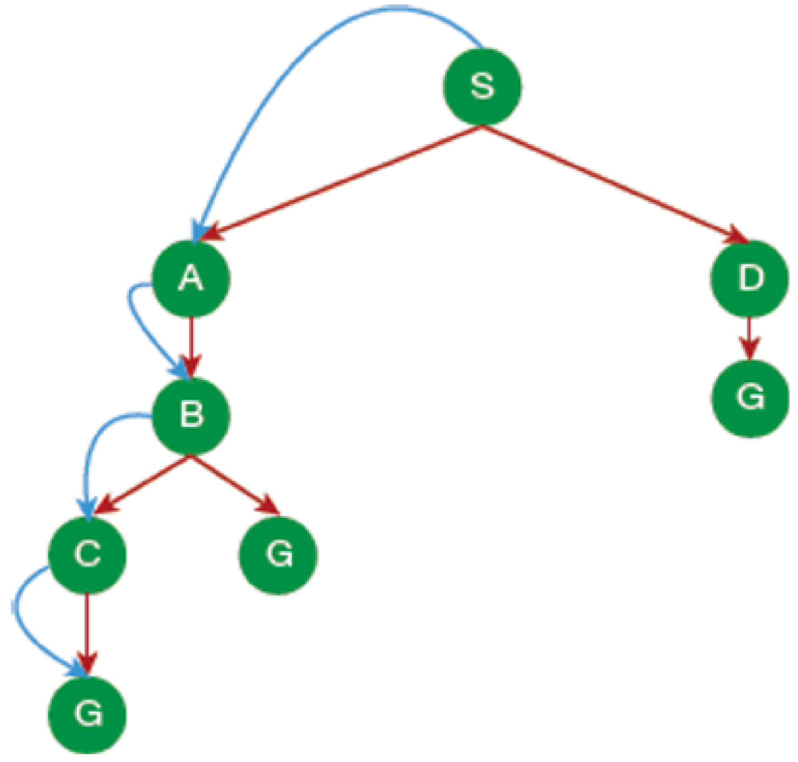
Module 1: Introduction to Artificial Intelligence

3.6 Uninformed Search Algorithms - 3.6.1 Depth First Search (DFS)

DFS Example – Finding Path to Node G

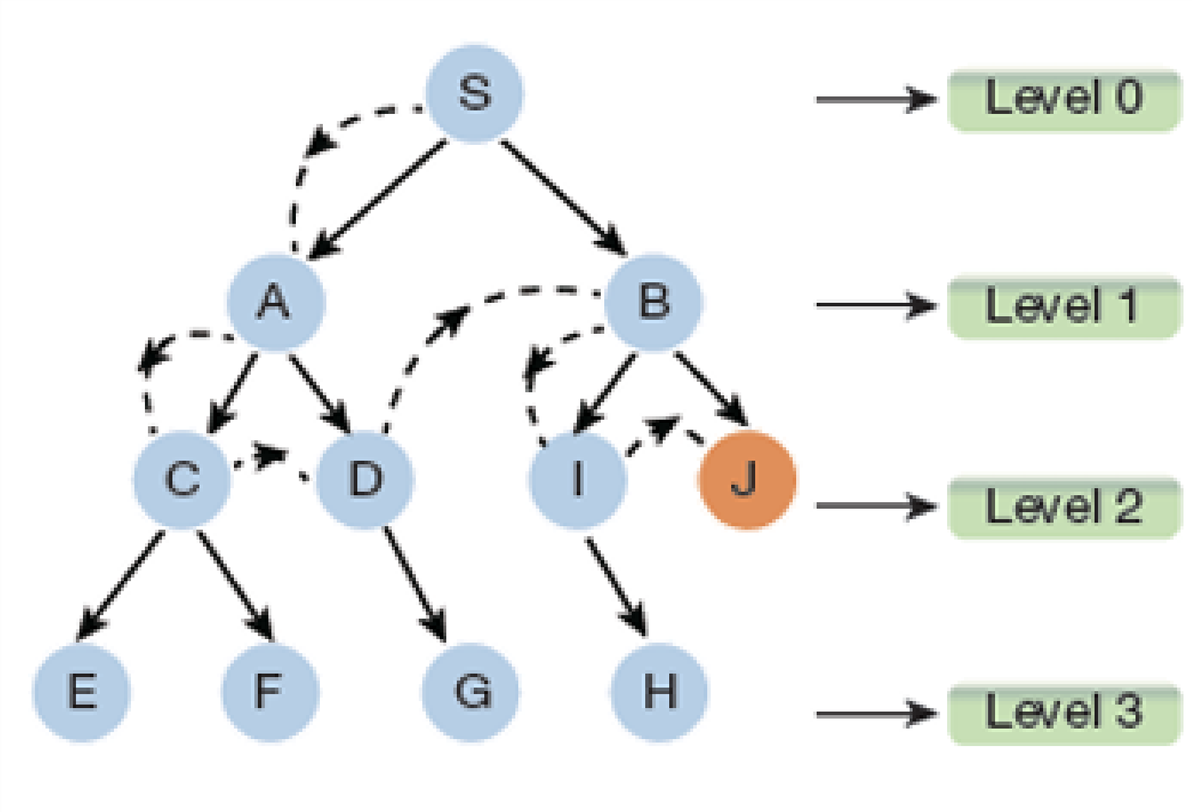


1. Start at node S, goal is node G.
2. How DFS Works:
   a. DFS (Depth-First Search) explores the deepest node first before backtracking.
   b. It goes down one branch completely until it reaches the goal or the end of that branch.
3. Step-by-Step Traversal:
   a. Start at S → go to A
   b. From A → go to B
   c. From B → go to C
   d. From C → reach G (goal found)
4. **Path Found**: S → A → B → C → G
5. The traversal is shown in blue arrows



3.6 Uninformed Search Algorithms - 3.6.2 Depth-Limited Search (DLS)

Depth-Limited Search (DLS) is similar to Depth First Search (DFS), but with a predetermined depth limit to avoid infinite paths. When the search reaches the specified depth limit, nodes at that depth are treated as leaf nodes (i.e., nodes with no successors).



Termination Conditions of DLS:

1. No Solution: If the problem has no solution, it's called standard error failure.
2. No Solution within Limit: If the solution is not found within the given depth limit, it's called cut-off failure.
3. Solution Found: If the solution is found within the depth limit, the algorithm stops.

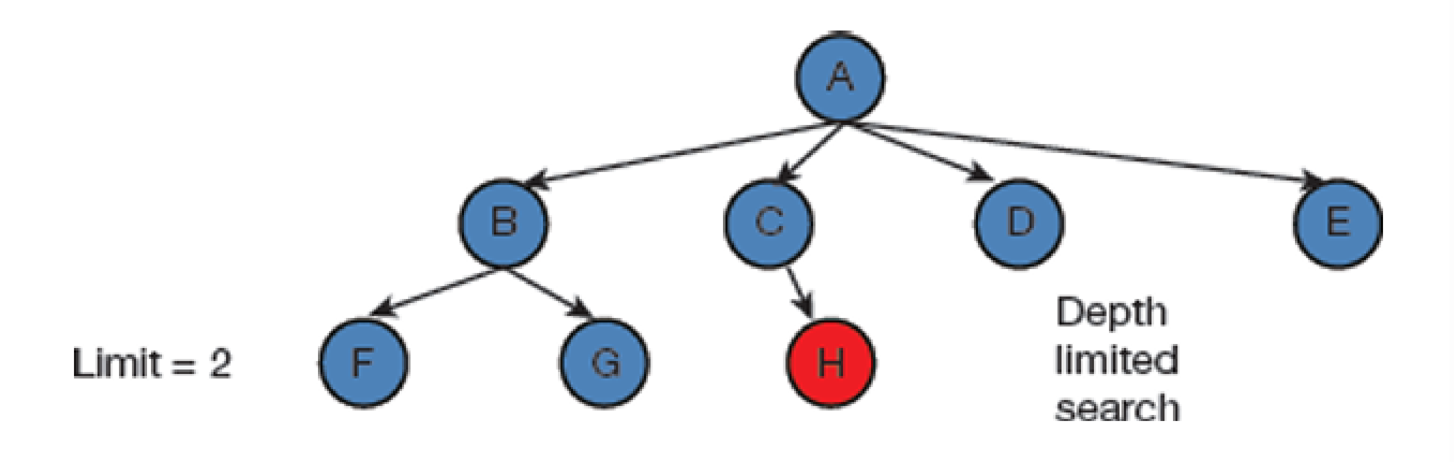
Example:

- If the depth limit is set to 2, the algorithm will not explore level 3 of the tree, so nodes E, F, G, H won't be traversed.

3.6 Uninformed Search Algorithms - 3.6.2 Depth-Limited Search (DLS)

DLS Example – Searching for Node H (Limit = 2)

- Search for node H using Depth-Limited Search (DLS) with a depth limit of 2.
- **How DLS Works:**
  - Start at root node A (level 0).
  - Explore all nodes at level 1: B, C, D, E.
  - Move to level 2 and explore children of B → H not found → backtrack.
  - Explore children of C at level 2 → H found.
- **Traversal Path:**
  - A → B → (children of B checked) → backtrack
  - A → C → H → goal found



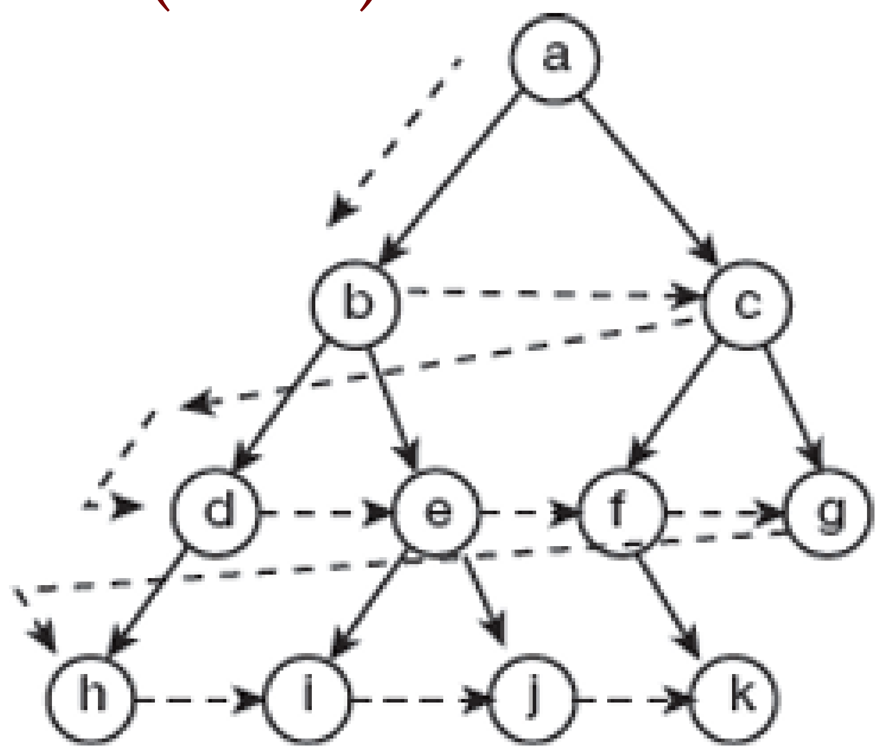
- **Key Idea:**
  - DLS explores nodes up to a fixed depth limit.
  - If the goal isn't found at the current depth, it backtracks to explore other branches.
  - Useful when the goal depth is known or limited.

3.6 Uninformed Search Algorithms - 3.6.3 Breadth First Search (BFS)

Breadth First Search (BFS) is an algorithm that traverses the tree breadthwise, meaning it explores all nodes at the current level before moving to the next level.

How It Works:

1. Start at the root node: The search begins at the root node (node A).
2. Traverse level by level: First, visit the immediate children of the root (nodes B and C).
3. Move to the next level: After visiting all nodes at the current level, the search moves to the next level, visiting nodes D, E, F, and G.
4. Continue level-wise traversal: The algorithm continues this process, exploring all neighbor nodes (children) at each level before moving deeper.



Example

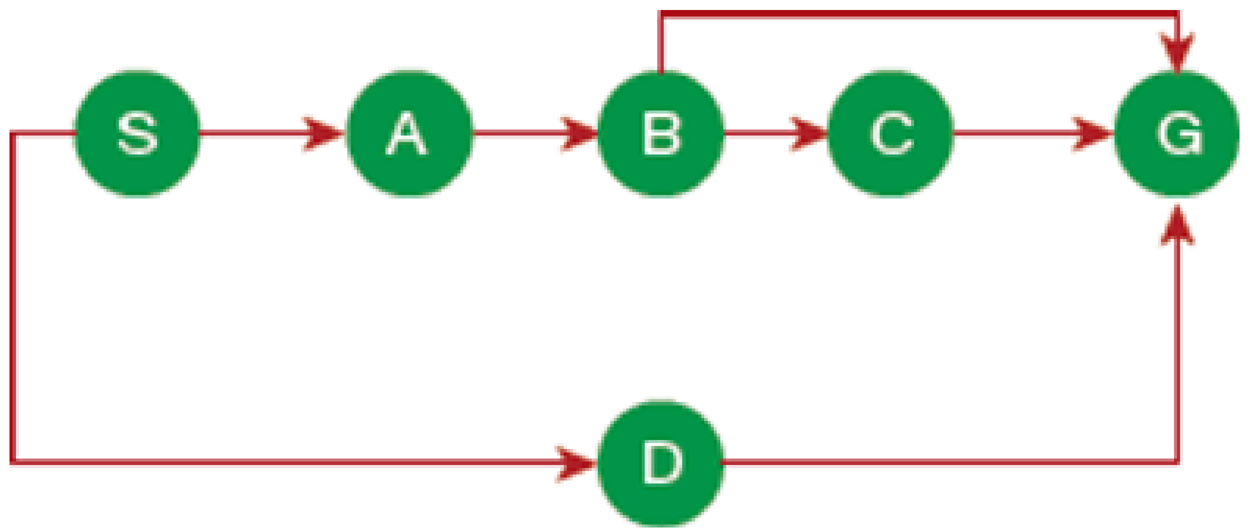
- Start at node A.
- Visit its children: nodes B and C.
- After visiting nodes at level 1, move to level 2: nodes D, E, F, and G.
- Then, visit level 3: nodes H to K.

3.6 Uninformed Search Algorithms - 3.6.3 Breadth First Search (BFS)

Example: Find the BFS traversal from node S to node G

Find the path from S to G using Breadth-First Search (BFS).

- **How BFS Works:**
  - BFS explores nodes level by level, starting from the root (shallowest nodes first).
  - It explores all nodes at the current depth before moving to the next level.
- **Step-by-Step Traversal:**
  - Start at S → explore children nodes at level 1.
  - Node D is explored first → then G is reached.



- **Path Found:**
  - S → D → G
- **Key Idea:**
  - BFS always finds the shallowest solution (fewest number of nodes in path).
  - Path length = level of the shallowest solution.

3.6 Uninformed Search Algorithms - 3.6.4 Uniform Cost Search (UCS)

UCS finds the cheapest path from a start node to a goal node when the step costs are different.

How It Works:

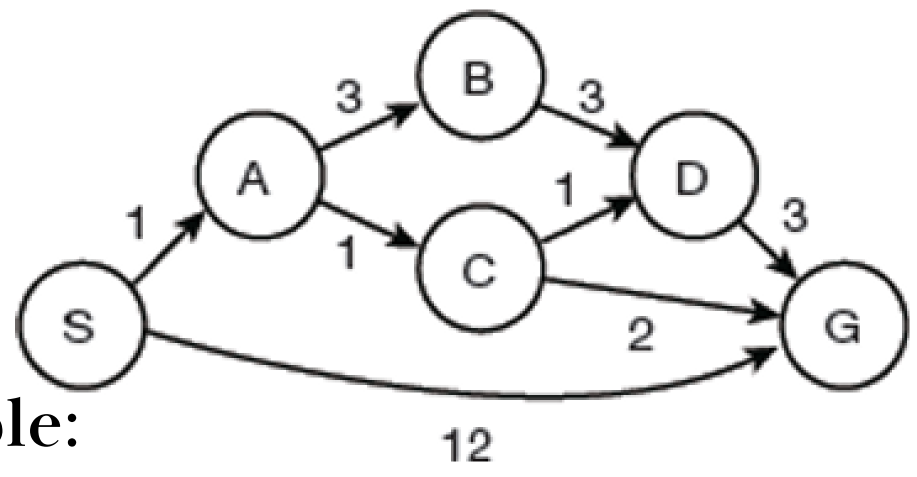
1. It calculates the total cost to reach each node from the start.
2. Always expands the node with the lowest cumulative cost next.
3. It does not follow depth-first or breadth-first order.

Cost of a Node:

- cost(node) = sum of costs from start to this node
- cost(start) = 0

Key Feature:

- UCS always finds the optimal (cheapest) path.
- Works like BFS if all step costs are the same.



Example:

- Start at S (start) and goal is G.
- If A has lower cost than G, UCS will explore A first.
- Then it compares children (B and C), chooses the one with lowest cost (say C), and continues until reaching G.

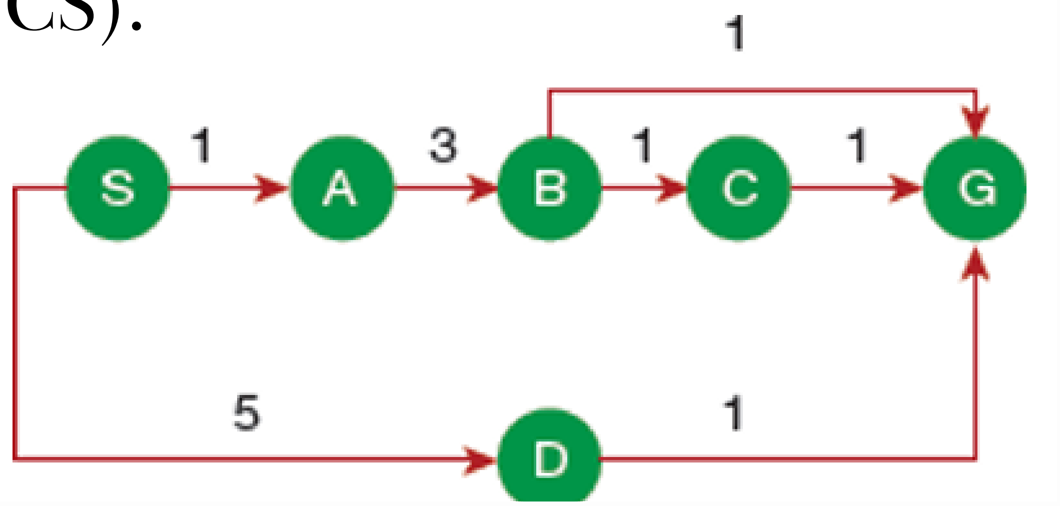
Implementation:

- Usually uses a priority queue, which gives priority to nodes with lowest cumulative cost.

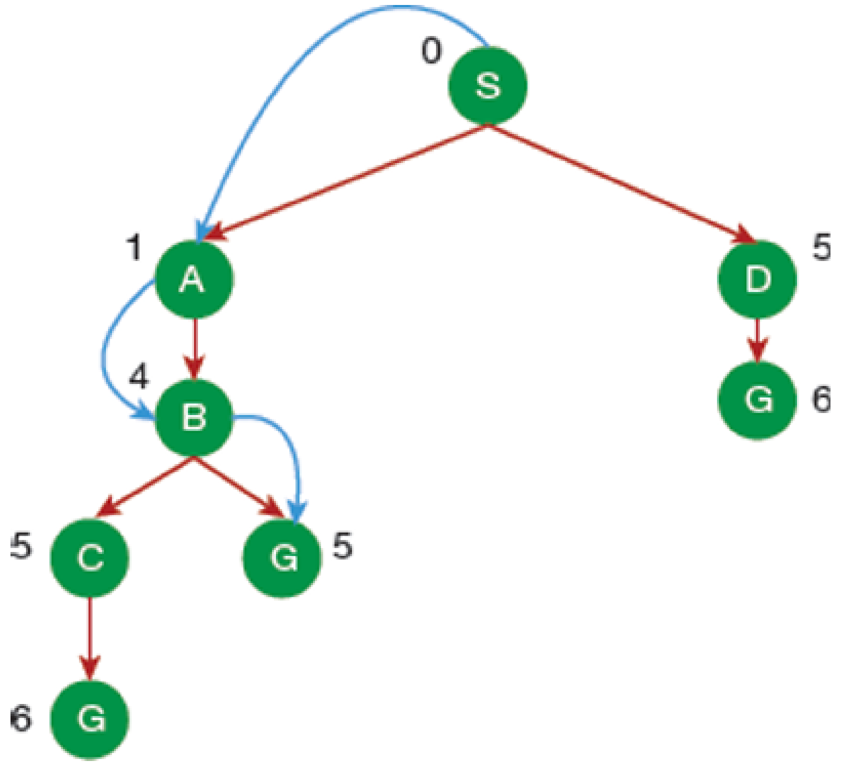
3.6 Uninformed Search Algorithms - 3.6.4 Uniform Cost Search (UCS)

Example: Find the path and cost to move from node S to node G in the graph given below

Find the cheapest path from S (start) to G (goal) using Uniform Cost Search (UCS).



- **How UCS Works:**
  - UCS expands nodes based on cumulative cost from the start node.
  - Always selects the node with the lowest total cost to explore next.
- **Step-by-Step Traversal:**
  - Start at S → check child nodes A and other options.
  - Choose A because it has the lowest cumulative cost.
  - From A, explore B → cost is still lowest.
  - From B, reach G → goal reached with least total cost.
- **Path Found:**
  - S → A → B → G
- **Total Cost:**
  - 5
- Key Idea: UCS always finds the optimal path based on actual costs, not depth or heuristic.



3.6 Uninformed Search Algorithms - 3.6.5 Iterative Deepening Depth-First Search (IDDFS)

- IDDFS is used to search for a goal node when the depth of the goal is unknown.
- It combines the advantages of DFS (low memory use) and BFS (guaranteed shortest path).

How It Works:

1. Start with a depth limit of 1, perform DFS up to that depth.
2. If the goal isn't found, increase the depth limit by 1 and repeat DFS.
3. Continue increasing the depth limit until the goal is found.

Key Feature:

- Memory Efficient: Only stores nodes in the current DFS path (like DFS).
- Fast Search: Gradually explores all levels (like BFS).
- Does not generate nodes beyond the current depth limit until needed.

Best Used When:

- The search space is large.
- The depth of the goal node is unknown.

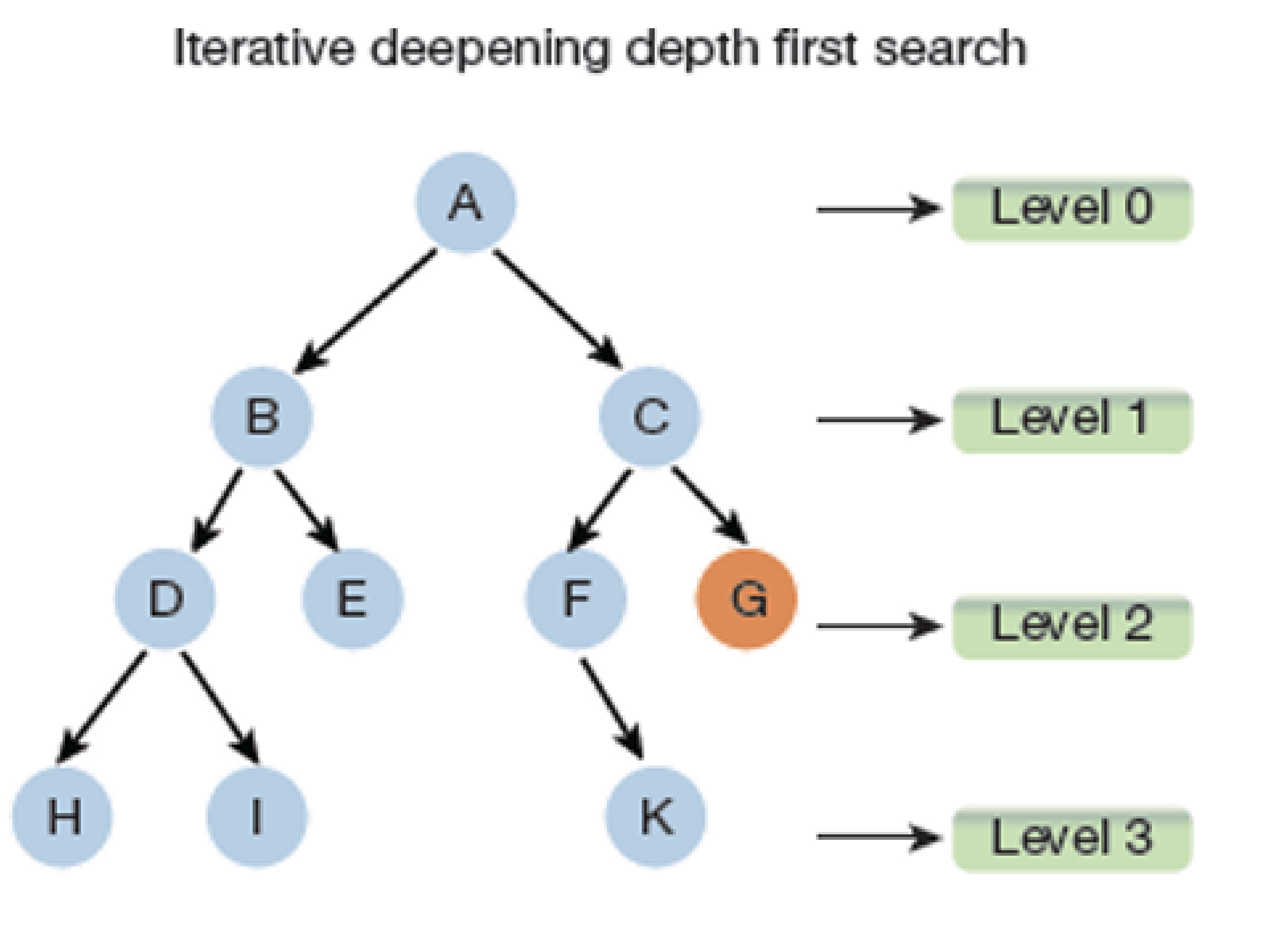
3.6 Uninformed Search Algorithms - 3.6.5 Iterative Deepening Depth-First Search (IDDFS)

Example: Traverse the given tree using the iterative deepening depth-first search algorithm.

Search for a goal node H using Iterative Deepening Depth-First Search (IDDFS).

- **How IDDFS Works:**
  - Performs DFS repeatedly with gradually increasing depth limits.
  - Each iteration explores all nodes up to the current depth limit.

- **Step-by-Step Traversal:**
  - Iteration 1 (Depth 0): Explore node A
  - Iteration 2 (Depth 1): Explore nodes B and C
  - Iteration 3 (Depth 2): Explore nodes D, E, F, G
  - Iteration 4 (Depth 3): Explore node H → goal found



3.6 Uninformed Search Algorithms - 3.6.5 Iterative Deepening Depth-First Search (IDDFS)

Example: Traverse the given tree using the iterative deepening depth-first search algorithm.

Key Features:

- Memory Efficient: Only stores nodes along the current DFS path.
- Complete: Guarantees to find the goal if it exists.
- Combines DFS depth efficiency with BFS completeness.

Complexity:

- Time Complexity: O(b^d)
- Space Complexity: O(b·d)
  - Where b = branching factor, d = depth of the goal

Key Idea:

- IDDFS gradually deepens the DFS limit until the goal is reached, balancing memory efficiency and completeness.

3.6 Uninformed Search Algorithms - 3.6.6 Bidirectional Search

To find the shortest path between a start node and a goal node more efficiently.

How It Works:

1. The search happens from both directions at the same time:
   a. **Forward Search:** From the start node toward the goal.
   b. **Backward Search:** From the goal node toward the start.
2. The search stops when the two paths meet at a common node.

Advantage:

- Reduces the search space because each search only goes half the total distance.
- Faster than searching in a single direction.

Key Idea:

- Instead of one big search, two smaller searches meet in the middle, saving time and memory.

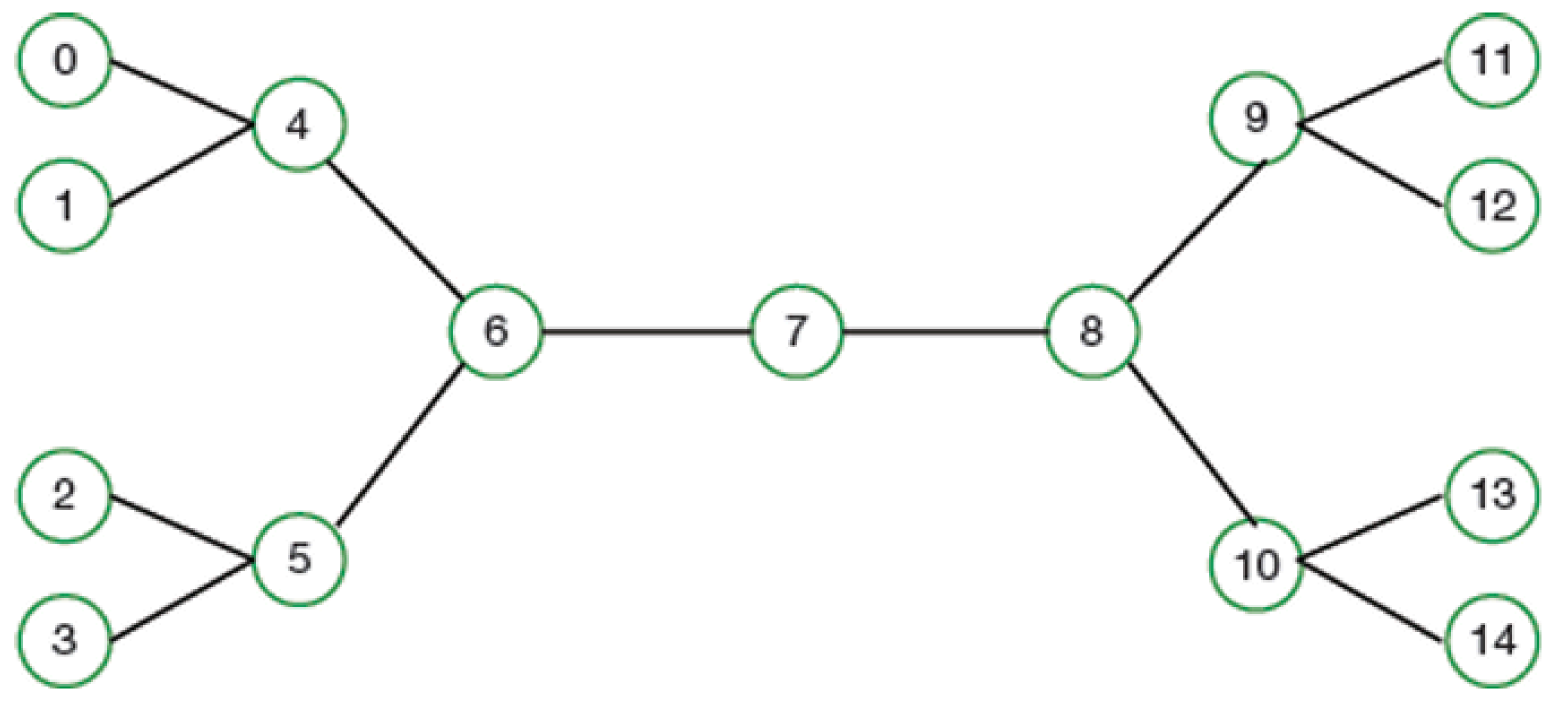
3.6 Uninformed Search Algorithms - 3.6.6 Bidirectional Search

Example: Consider the graph given below and apply bidirectional search on it to reach goal node 14 from source node 0.

Find the path from source node 0 to goal node 14 using Bidirectional Search.

How Bidirectional Search Works:

- Two searches are run simultaneously:
  - Forward Search: From 0 → 14
  - Backward Search: From 14 → 0
- Both searches continue until they meet at a common node.



Step-by-Step Traversal:

- Forward search: 0 → … → 7
- Backward search: 14 → … → 7
- Intersection at node 7 → path found

- **Path Found:**
- **Concatenate forward and backward paths:**
  - **0 → … → 7 → … → 14**

3.7 Informed Search Algorithms

To find the goal node efficiently using extra information about the search space.

- **Key Feature:**
  - Uses a heuristic function to estimate how close a node is to the goal.
- **How it Works:**
  - The heuristic function h(n) calculates the estimated cost from the current node n to the goal.
  - The agent uses this information to choose the most promising path, reducing unnecessary exploration.
- **Advantages:**
  - Explores fewer nodes than uninformed search (DFS, BFS).
  - Reaches the goal faster in large search spaces.
- **Guarantee:**
  - Heuristic may not always give the absolute best path, but it finds a good solution in reasonable time.
- **Key Idea:**
  - Informed search = smarter search because it “knows” which directions are likely better.

3.7 Informed Search Algorithms - 3.7.1 Pure Heuristic Search

To solve problems with a large number of possible states efficiently by using problem-specific knowledge.

- **Key Idea:**
  - Expands nodes based on their heuristic value h(n) (an estimate of distance to the goal).
- **How it Works:**
  - Maintains two lists:
    - OPEN list: Nodes yet to be expanded.
    - CLOSED list: Nodes already expanded.
  - In each step, the node with the lowest heuristic value is expanded first.
- **Process:**
  - Apply heuristic to child nodes.
  - Add child nodes to OPEN list based on their heuristic value.
  - Keep shorter paths, discard longer ones.
  - Repeat until goal state is reached.

- Benefit:
  - Reduces unnecessary exploration and focuses on the most promising paths.
  - Examples:
  - Greedy Best-First Search
  - A Search Algorithm*

3.7 Informed Search Algorithms - 3.7.2 Best-First Search Algorithm (Greedy Search)

To find the goal node quickly by always choosing the most promising path.

- **Key Idea:**
  - Uses a heuristic function h(n) to estimate which node is closest to the goal.
  - Combines advantages of DFS (deep exploration) and BFS (level-wise search).

- **How it Works:**
  a. Insert the start node in the OPEN list.
  b. If OPEN list is empty, stop (goal not found).
  c. Remove the node with lowest heuristic value h(n) from OPEN, add it to CLOSED list.
  d. Expand this node and generate its successor nodes.
  e. If any successor is the goal, stop and return success.
  f. If a successor is not in OPEN or CLOSED, add it to OPEN list.
  g. Repeat from Step b until goal is found.

Module 1: Introduction to Artificial Intelligence

3.7 Informed Search Algorithms - 3.7.2 Best-First Search Algorithm (Greedy Search)

- **Implementation:**
  - Usually uses a priority queue, giving priority to nodes with lowest heuristic value.
- **Benefit:**
  - Finds the goal faster by focusing on promising paths rather than exploring all nodes.
- **Key Idea:**
  - Greedy because it always picks the node that looks best now, but it may not always find the shortest path.
-

3.7 Informed Search Algorithms - 3.7.2 Best-First Search Algorithm (Greedy Search)

Example: Consider the tree given below and traverse it using greedy best first search algorithm.

- Traverse the tree from node S to find the goal node using the Greedy Best-First Search Algorithm.
- The algorithm uses the heuristic value h(n) to choose the most promising node at each step.

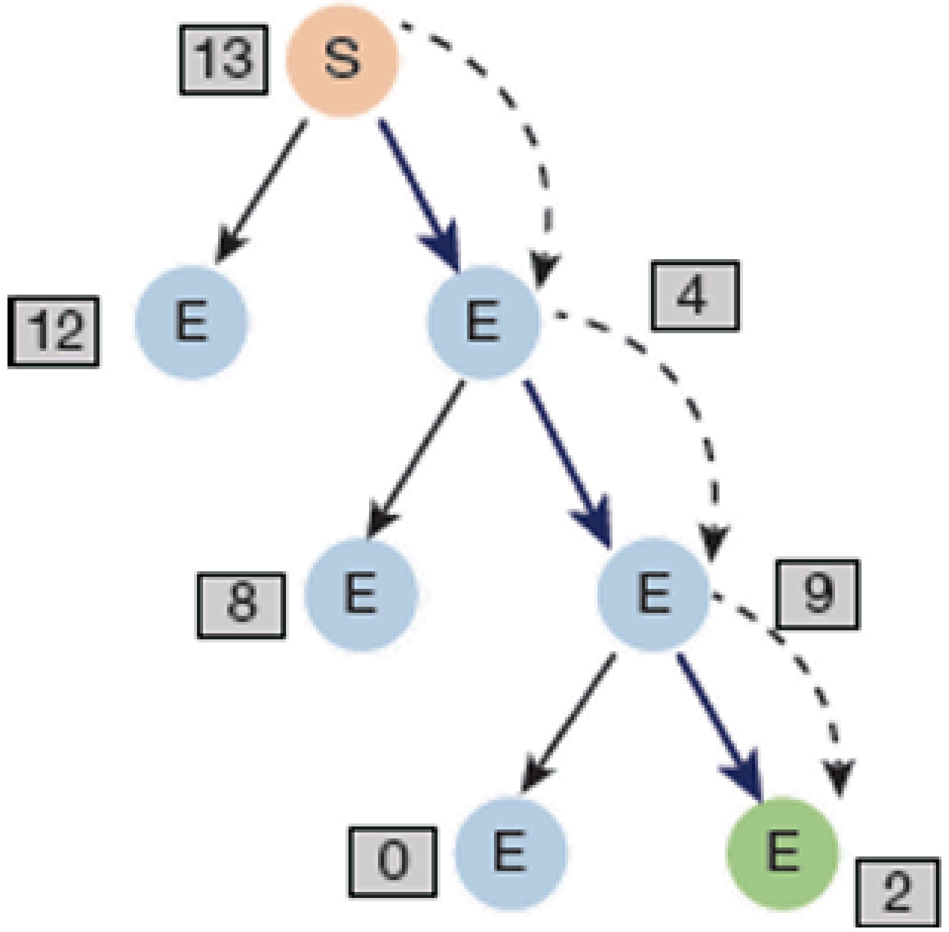


Table shows the heuristic estimates:

Node	h(n)	Node	h(n)	Node	h(n)
A	11	E	4	I, J	3
B	5	F	2	S	15
C, D	9	H	7	G	0

3.7 Informed Search Algorithms - 3.7.2 Best-First Search Algorithm (Greedy Search)

Example: Consider the tree given below and traverse it using greedy best first search algorithm.

- How Greedy Best-First Search Works:
  - At each step, the node with the lowest heuristic value (h(n)) is expanded.
  - The OPEN list contains nodes to be expanded, and the CLOSED list contains nodes already expanded.

3.7 Informed Search Algorithms - 3.7.2 Best-First Search Algorithm (Greedy Search)

Example: Consider the tree given below and traverse it using greedy best first search algorithm.

- Step-by-Step Traversal:
  - **Iteration 1:**
  - Start at S.
  - Successors of S: A, B.
  - OPEN = [A, B], CLOSED = [S].
  - B has the lowest h(n) value, so expand B.
  - OPEN = [A], CLOSED = [S, B].

- Step-by-Step Traversal:
  - **Iteration 2:**
  - Generate successors of B: E, F, A.
  - OPEN = [E, F, A], CLOSED = [S, B].
  - F has the lowest h(n) value, so expand F.
  - OPEN = [E, A], CLOSED = [S, B, F].

3.7 Informed Search Algorithms - 3.7.2 Best-First Search Algorithm (Greedy Search)

Example: Consider the tree given below and traverse it using greedy best first search algorithm.

- **Iteration 3:**
- Generate successors of F: I, G, E, A.
- OPEN = [I, G, E, A], CLOSED = [S, B, F].
- G is the goal, so the algorithm terminates successfully.
- OPEN = [I, E, A], CLOSED = [S, B, F, G].
  -

- Path Found:
  - The path is S → B → F → G.
- Key Idea:
  - Greedy Search picks the node that looks best now, but it may not always find the shortest path.
  - It's not optimal and can behave incompletely if the heuristic is not well-designed.

Module 1: Introduction to Artificial Intelligence

1. Introduction to Artificial Intelligence

1. Definition of Artificial Intelligence
2. How Does AI Work?
3. Advantages and Disadvantages of Artificial Intelligence
4. History of Artificial Intelligence
5. Types of Artificial Intelligence:
   a. Weak AI vs. Strong AI
   b. Reactive Machines
   c. Limited Memory
   d. Theory of Mind
   e. Self-Awareness
6. Is Artificial Intelligence the Same as Augmented Intelligence and Cognitive Computing?
7. Introduction to Machine Learning and Deep Learning

2. Machine Intelligence

1. Defining Intelligence
2. Components of Intelligence
3. Differences Between Human and Machine Intelligence
4. Agent and Environment in AI
5. Search Algorithms:
   a. Uninformed Search Algorithms
   b. Informed Search Algorithms:
      i. Pure Heuristic Search
      ii. Best-First Search Algorithm (Greedy Search)

Thank You

Module 1 - Completed