

# Artificial Intelligence

## Suggestions

### **What is Artificial Intelligence?**

Artificial Intelligence is composed of two words Artificial and **Intelligence**, where Artificial defines "*man-made*," and intelligence defines "*thinking power*", hence AI means "*a man-made thinking power*." It is a branch of computer science by which we can create intelligent machines which can behave like a human, think like humans, and able to make decisions.

### **How does artificial intelligence work? explain briefly.**

Artificial intelligence (AI) works by using algorithms and models to simulate aspects of human cognition. Here are the key components:

1. **Data Input:** AI systems require large amounts of data to learn from. This data can be anything from text, images, and videos to sensor readings and user interactions.
2. **Learning Algorithms:** AI employs various algorithms to analyse and learn patterns from the data. These algorithms can be categorized into:
  - **Supervised Learning:** The AI is trained on labelled data, meaning each input comes with a corresponding output. The system learns to predict the output from new inputs.
  - **Unsupervised Learning:** The AI finds patterns and relationships in data without labelled outcomes, often used for clustering or dimensionality reduction.
  - **Reinforcement Learning:** The AI learns by interacting with its environment and receiving rewards or penalties, optimizing its actions to maximize cumulative rewards.
3. **Model Training:** The AI model is trained using the chosen learning algorithm. During training, the model adjusts its parameters to minimize errors in predictions or to better identify patterns.

4. **Inference:** Once trained, the AI model can make predictions or decisions based on new, unseen data. This is the practical application phase, where the AI performs tasks like recognizing objects in images, understanding spoken language, or recommending products.
5. **Feedback and Improvement:** AI systems can improve over time through continuous learning and feedback, refining their models with new data and adjusting their algorithms to enhance performance.

In summary, AI works by using data and sophisticated algorithms to learn, make predictions, and adapt over time, simulating intelligent behaviour to perform tasks that typically require human intelligence.

### **How Artificial Intelligence is being used in everyday life?**

Artificial intelligence is integrated into various aspects of everyday life, enhancing convenience, efficiency, and personalization. Here are some common uses:

1. **Smartphones:** AI powers virtual assistants like Siri, Google Assistant, and Alexa, enabling voice commands, smart replies, and predictive text.
2. **Social Media:** AI algorithms personalize content feeds, recommend friends, and filter out inappropriate content.
3. **Online Shopping:** E-commerce platforms use AI for personalized recommendations, dynamic pricing, and customer service chatbots.
4. **Streaming Services:** Platforms like Netflix and Spotify use AI to suggest movies, shows, and music based on user preferences and behavior.
5. **Smart Homes:** AI is embedded in smart home devices like thermostats, security cameras, and lighting systems, enabling automation and remote control.
6. **Healthcare:** AI assists in diagnostic tools, personalized treatment plans, and health monitoring through wearable devices.
7. **Transportation:** AI is used in navigation apps for real-time traffic updates, route optimization, and autonomous driving technology.
8. **Finance:** AI helps in fraud detection, credit scoring, personalized financial advice, and algorithmic trading.

These applications showcase AI's ability to enhance user experiences, improve decision-making, and streamline various daily activities.

### **Explain the goals of Artificial Intelligence in brief.**

The goals of artificial intelligence (AI) are centred around creating systems that can perform tasks that typically require human intelligence. Here are the primary objectives:

1. **Automation:** Developing systems that can automate repetitive and mundane tasks, increasing efficiency and reducing human labour.
2. **Problem Solving:** Creating algorithms that can solve complex problems, making decisions, and providing solutions in various fields such as healthcare, finance, and logistics.
3. **Learning and Adaptation:** Building models that can learn from data, adapt to new information, and improve their performance over time.
4. **Natural Interaction:** Enabling machines to understand and interact with humans naturally, including language processing, speech recognition, and emotional understanding.
5. **Perception and Understanding:** Equipping systems with the ability to perceive their environment through sensors, and understand visual, auditory, and other sensory data.
6. **Knowledge Representation:** Developing methods to represent knowledge about the world in a form that a computer system can utilize to solve complex tasks.
7. **Generalization:** Creating AI that can generalize learning from one task to another, aiming for versatility similar to human cognitive abilities.

These goals collectively aim to create intelligent systems that can augment human capabilities, enhance productivity, and contribute to advancements across various domains.

### **What is the foundational concept of Artificial Intelligence?**

The foundational concept of Artificial Intelligence is to create machines that can perform tasks that would typically require human-level intelligence. This includes tasks such as decision-making, natural language processing, pattern recognition, and learning from experience. The goal is to create intelligent machines that can

mimic, and in some cases surpass, human intelligence in order to solve complex problems and improve our ability to automate various processes. The foundational concept of AI involves developing algorithms and techniques that enable machines to process and interpret large amounts of data, learn from patterns, and make decisions based on that learning.

What are some of the key components of an AI system?

Key components of an AI system include:

1. **Data:** Large volumes of relevant data for training and improving the AI models.
2. **Algorithms:** Methods for processing data and making decisions, including machine learning algorithms.
3. **Models:** Trained algorithms that can make predictions or perform specific tasks based on new data.
4. **Computing Power:** Hardware and infrastructure required to process large datasets and complex computations.
5. **Training:** The process of feeding data into the algorithms to help the AI system learn and improve its performance.
6. **Inference:** The ability of the trained model to apply its knowledge to new, unseen data and make decisions.
7. **Evaluation:** Techniques to measure the accuracy and effectiveness of the AI system and improve it over time.

What are some of the main subfields of Artificial Intelligence?

Some of the main subfields of artificial intelligence (AI) include:

1. **Machine Learning (ML):** Focuses on developing algorithms that enable systems to learn from and make decisions based on data.
2. **Natural Language Processing (NLP):** Involves the interaction between computers and human language, enabling machines to understand, interpret, and generate human language.

3. **Computer Vision:** Deals with enabling machines to interpret and make decisions based on visual data from the world, such as images and videos.
4. **Robotics:** Combines AI with mechanical engineering to create intelligent robots that can perform tasks autonomously.
5. **Expert Systems:** Uses AI to replicate the decision-making abilities of a human expert in specific domains, such as medical diagnosis or financial forecasting.
6. **Neural Networks:** Models inspired by the human brain that are used in deep learning to recognize patterns and perform complex tasks.
7. **Reinforcement Learning:** Focuses on training models to make sequences of decisions by rewarding desirable behaviours and penalizing undesirable ones.
8. **Knowledge Representation and Reasoning:** Involves methods for representing information about the world in a form that AI systems can utilize to solve complex problems.
9. **Speech Recognition:** Enables machines to understand and process human speech, allowing for voice control and interaction.
10. **Artificial General Intelligence (AGI):** Aiming to develop machines with the ability to understand, learn, and apply knowledge across a wide range of tasks, similar to human intelligence.

What are some of the challenges in developing Artificial Intelligence systems?

Some challenges in developing artificial intelligence (AI) systems include:

1. **Data Quality and Quantity:** Ensuring access to large, high-quality, and diverse datasets for training AI models.
2. **Computational Resources:** Providing sufficient computing power and infrastructure to process complex algorithms and large datasets.
3. **Algorithmic Bias:** Addressing and mitigating biases in AI algorithms that can lead to unfair or discriminatory outcomes.
4. **Interpretability:** Making AI models transparent and understandable to humans, especially in critical applications.
5. **Scalability:** Ensuring AI systems can operate efficiently at large scales and handle increasing amounts of data.

6. **Ethical Concerns:** Navigating the ethical implications of AI, including privacy, security, and the potential for misuse.
7. **Integration and Deployment:** Effectively integrating AI systems into existing workflows and ensuring they operate reliably in real-world conditions.
8. **Continuous Learning:** Developing AI systems that can adapt to new data and evolving environments without significant retraining.

What are some of the recent breakthroughs in AI research and how are they impacting the field?

Recent breakthroughs in AI research have significantly advanced the field and are impacting various industries. Here are a few notable developments:

1. **Transformer Models and NLP:** The development of transformer models like GPT-4 and BERT has revolutionized natural language processing, enabling more accurate language understanding, generation, and translation. These models have improved chatbots, virtual assistants, and automated content creation.
2. **Reinforcement Learning:** Advances in reinforcement learning, exemplified by AlphaGo and AlphaZero, have demonstrated the ability of AI to master complex games and strategic tasks. These techniques are being applied to robotics, autonomous vehicles, and resource management.
3. **Computer Vision:** Improved convolutional neural networks (CNNs) and techniques like generative adversarial networks (GANs) have enhanced image and video recognition, enabling applications in medical imaging, autonomous driving, and facial recognition.
4. **Multimodal AI:** Models that combine text, image, and audio data, such as CLIP and DALL-E, are enabling more comprehensive understanding and generation capabilities, impacting fields like creative arts, marketing, and education.
5. **Explainable AI (XAI):** Progress in explainability and interpretability of AI models helps address the "black box" problem, making AI decisions more

transparent and trustworthy, particularly in healthcare, finance, and legal sectors.

6. **Edge AI:** Advancements in edge computing allow AI models to run on local devices rather than centralized servers, leading to faster processing times and enhanced privacy for applications in IoT, smart homes, and mobile devices.
7. **Quantum Computing and AI:** Early explorations in integrating AI with quantum computing promise to solve problems currently infeasible for classical computers, potentially transforming cryptography, optimization, and complex simulations.

These breakthroughs are driving AI's integration into everyday life, enhancing capabilities across industries, improving efficiency, and enabling new applications that were previously thought impossible.

How are companies and organizations using AI to improve their operations and create new products and services?

Companies and organizations are leveraging AI to enhance their operations and innovate new products and services in several key ways:

1. **Automation:** Automating repetitive tasks and processes to increase efficiency, reduce errors, and lower operational costs. Examples include robotic process automation (RPA) in finance and customer service.
2. **Personalization:** Offering personalized recommendations and experiences to customers by analysing data on preferences and behaviour, as seen in e-commerce, streaming services, and marketing.
3. **Predictive Analytics:** Using AI to analyse historical data and predict future trends, enabling better decision-making in supply chain management, financial forecasting, and inventory management.
4. **Customer Support:** Implementing AI-powered chatbots and virtual assistants to provide 24/7 customer support, handle inquiries, and resolve issues quickly, improving customer satisfaction.
5. **Quality Control:** Enhancing quality control and defect detection in manufacturing through computer vision and machine learning, leading to higher product standards and reduced waste.

6. **Healthcare:** Developing AI-driven diagnostic tools, personalized treatment plans, and predictive models to improve patient outcomes and optimize hospital operations.
7. **Fraud Detection:** Employing AI to detect and prevent fraud in real-time by analysing transaction patterns and identifying anomalies in banking and insurance.
8. **Product Innovation:** Creating new products and services powered by AI, such as autonomous vehicles, smart home devices, and AI-driven financial advisory services.
9. **Employee Productivity:** Enhancing employee productivity and decision-making through AI tools that provide insights, automate administrative tasks, and facilitate collaboration.

These applications of AI are helping organizations improve efficiency, reduce costs, enhance customer experiences, and drive innovation across various sectors.

### **What are some of the ethical considerations associated with the development and use of AI systems?**

Ethical considerations associated with the development and use of AI systems include:

1. **Bias and Fairness:** Ensuring AI systems do not perpetuate or exacerbate biases present in training data, which can lead to unfair or discriminatory outcomes.
2. **Privacy:** Protecting user data from unauthorized access and misuse, and ensuring transparency about data collection and usage practices.
3. **Transparency and Explainability:** Making AI decision-making processes understandable and transparent to users, especially in critical applications like healthcare and criminal justice.
4. **Accountability:** Establishing clear lines of responsibility for AI actions and decisions, and ensuring there are mechanisms for recourse if AI systems cause harm.
5. **Security:** Safeguarding AI systems from cyberattacks and ensuring they do not become tools for malicious activities.
6. **Job Displacement:** Addressing the potential impact of AI on employment, including job displacement and the need for retraining and reskilling workers.



7. **Autonomy and Control:** Ensuring humans retain control over AI systems, particularly in high-stakes areas such as military and law enforcement.
8. **Misinformation:** Preventing the misuse of AI for generating and spreading false information or deepfakes, which can erode trust and cause societal harm.

These considerations are crucial for developing AI systems that are ethical, trustworthy, and beneficial to society.

### **What are the different types of AI?**

Different types of AI include:

1. **Narrow AI (Weak AI):** AI designed for specific tasks or domains, such as image recognition or language translation.
2. **Artificial General Intelligence (AGI):** AI with human-level cognitive abilities, capable of understanding, learning, and applying knowledge across a wide range of tasks.
3. **Superintelligent AI:** Hypothetical AI that surpasses human intelligence across all domains, potentially posing existential risks or providing unprecedented benefits.

These types represent a spectrum of AI capabilities, from specialized applications to broader, more general intelligence.

### **What is an agent in AI?**

In AI, an agent refers to any entity that perceives its environment through sensors and acts upon that environment through effectors to achieve specific goals. Agents can range from simple programs to complex systems, and they can be autonomous or controlled by humans. They are central to the study of intelligent systems and decision-making processes.

### **What are the types of agents?**

In AI, agents can be categorized into several types based on their characteristics and behaviour. Some common types of agents include:

1. **Simple Reflex Agents:** These agents select actions based solely on the current percept, without considering past perceptions or future consequences.

2. **Model-Based Reflex Agents:** These agents maintain an internal model of the world and use it to make decisions, taking into account both current and past perceptions.
3. **Goal-Based Agents:** These agents have predefined goals and take actions that maximize the likelihood of achieving those goals, often requiring planning and reasoning.
4. **Utility-Based Agents:** These agents make decisions based on the expected utility or value of different actions, aiming to maximize overall utility over time.
5. **Learning Agents:** These agents improve their performance over time by learning from experience, either through reinforcement learning, supervised learning, or unsupervised learning.
6. **Adaptive Agents:** These agents can adapt their behaviour and strategies in response to changes in the environment or user preferences.

These types of agents provide a framework for understanding and designing intelligent systems with varying degrees of complexity and autonomy.

### **Discuss about environment for agent.**

In the context of artificial intelligence, an environment refers to the external context or surroundings within which an agent operates and interacts. The environment defines the set of possible states, actions, and outcomes that the agent can perceive and affect. Here are some key points about the environment for an agent:

1. **State:** The state of the environment represents all the relevant information about the current situation or configuration. It includes variables such as the positions of objects, sensor readings, and any other relevant data that the agent can perceive.
2. **Actions:** Actions are the set of possible moves or operations that the agent can perform to affect the environment. The choice of action depends on the agent's goals, its current state, and any constraints imposed by the environment.
3. **Perception:** The agent perceives the state of the environment through sensors or observation mechanisms. These sensors provide information about the current state, which the agent uses to make decisions and choose actions.

4. **Feedback:** After taking an action, the agent receives feedback from the environment in the form of rewards, penalties, or changes in the state. This feedback helps the agent learn and adjust its behavior over time.
5. **Dynamicity:** Environments can be static or dynamic, meaning they may change over time due to the agent's actions or external factors. Dynamic environments pose additional challenges for agents, requiring them to adapt and plan accordingly.
6. **Determinism vs. Stochasticity:** Environments can be deterministic, where the outcome of actions is entirely predictable based on the current state, or stochastic, where outcomes are probabilistic and uncertain.

Understanding the environment is crucial for designing effective agents, as it directly influences the agent's decision-making process and performance. Different environments may require different strategies and algorithms to achieve the agent's goals successfully.

### **What is percept sequence?**

A percept sequence refers to the sequential stream of perceptual inputs or observations received by an agent from its environment over time. Each percept in the sequence represents the agent's perception of the environment at a specific point in time, providing information about the state of the environment, changes, or events occurring within it. By analysing and processing percept sequences, agents can make decisions, take actions, and adapt their behaviour to achieve their goals within the dynamic environment.

### **What is agent system?**

An agent system refers to a collection of agents working together within a shared environment to achieve common goals or objectives. Each agent within the system operates autonomously, perceiving its environment, making decisions, and taking actions based on its own goals and the information it receives. The agents may communicate with each other, collaborate, or compete to accomplish tasks more efficiently or effectively. Agent systems are commonly used in various domains, including robotics, multi-agent simulations, distributed systems, and artificial intelligence applications.

## **Define sensors, effectors, and actuators.**

**Sensor:** Sensor is a device which detects the change in the environment and sends the information to other electronic devices. An agent observes its environment through sensors.

**Actuators:** Actuators are the component of machines that converts energy into motion. The actuators are only responsible for moving and controlling a system. An actuator can be an electric motor, gears, rails, etc.

**Effectors:** Effectors are the devices which affect the environment. Effectors can be legs, wheels, arms, fingers, wings, fins, and display screen.

## **Write a short note on Intelligent Agent.**

An intelligent agent is an autonomous entity which act upon an environment using sensors and actuators for achieving goals. An intelligent agent may learn from the environment to achieve their goals. Driverless Car, thermostat are examples of an intelligent agent.

Following are the main four rules for an AI agent:

- **Rule 1:** An AI agent must have the ability to perceive the environment.
- **Rule 2:** The observation must be used to make decisions.
- **Rule 3:** Decision should result in an action.
- **Rule 4:** The action taken by the AI agent must be a rational. Rational actions are actions that maximize performance and yield the best positive outcome.

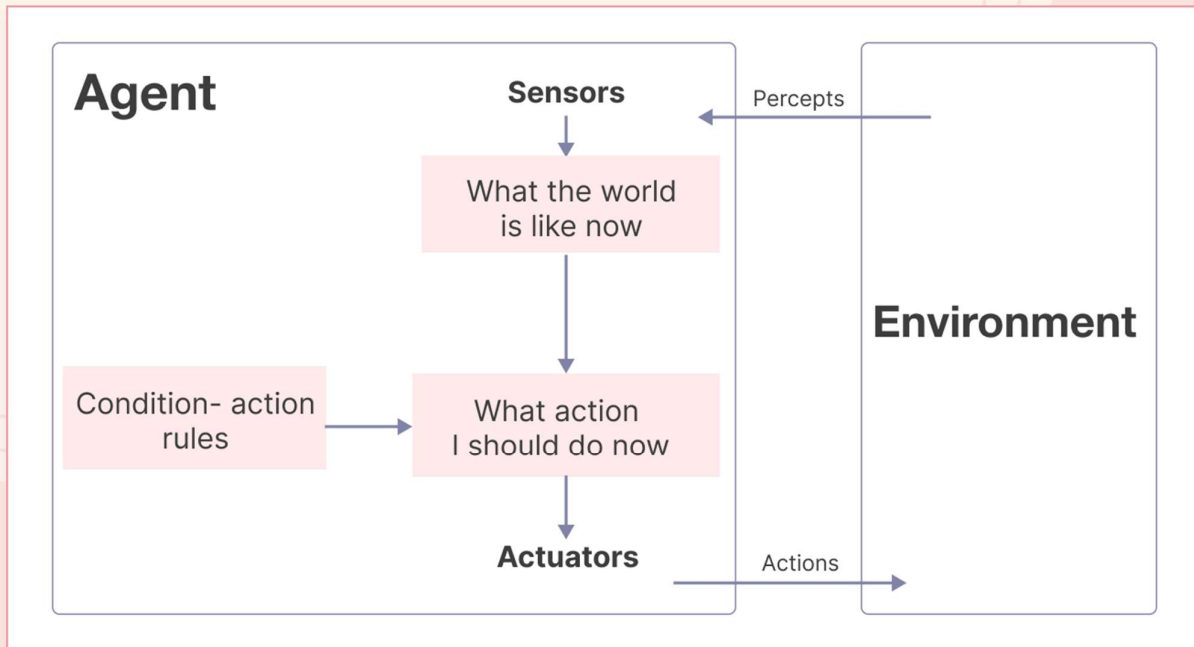
## **Describe different types of AI agents.**

### **1. Simple Reflex Agent**

A simple reflex agent is an AI system that follows pre-defined rules to make decisions. It only responds to the current situation without considering the past or future ramifications.

A simple reflex agent is suitable for environments with stable rules and straightforward actions, as its behaviour is purely reactive and responsive to immediate environmental changes.

# Simple Reflex Agent



 SIMFORM

## How does it work?

A simple reflex agent executes its functions by following the condition-action rule, which specifies what action to take in a certain condition.

## Example

A rule-based system developed to support automated customer support interactions. The system can automatically generate a predefined response containing instructions on resetting the password if a customer's message contains keywords indicating a password reset.

## Advantages of simple reflex agents

- Easy to design and implement, requiring minimal computational resources
- Real-time responses to environmental changes
- Highly reliable in situations where the sensors providing input are accurate, and the rules are well designed
- No need for extensive training or sophisticated hardware

## Limitations of simple reflex agents

Here are the limitations of the simple reflex agent:

- Prone to errors if the input sensors are faulty or the rules are poorly designed
- Have no memory or state, which limits their range of applicability
- Unable to handle partial observability or changes in the environment they have not been explicitly programmed for
- Limited to a specific set of actions and cannot adapt to new situations

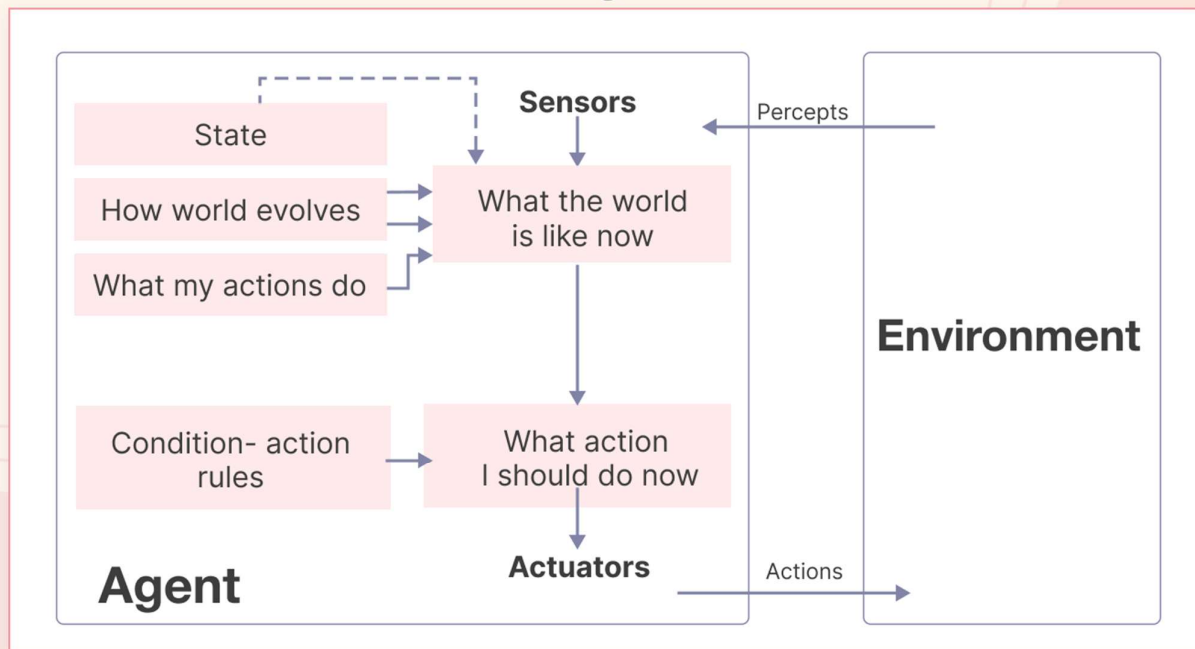
## 2. Model-based Reflex Agent

A model-based reflex performs actions based on a current percept and an internal state representing the unobservable world. It updates its internal state based on two factors:

- How the world evolves independently of the agent
- How does the agent's action affect the world

A cautious model-based reflex agent is a variant of a model-based reflex agent that also considers the possible consequences of its actions before executing them.

## Model-Based Reflex Agent



**How does it work?**

A model-based reflex agent follows the condition-action rule, which specifies the appropriate action to take in a given situation. But unlike a simple reflex agent, a model-based agent also employs its internal state to assess the condition during the decision and action process.

The model-based reflex agent operates in four stages:

1. **Sense:** It perceives the current state of the world with its sensors.
2. **Model:** It constructs an internal model of the world from what it sees.
3. **Reason:** It uses its model of the world to decide how to act based on a set of predefined rules or heuristics.
4. **Act:** The agent carries out the action that it has chosen.

### Example

One of the finest examples of a cautionary model-based reflex agent is [Amazon Bedrock](#).

Amazon Bedrock is a service that uses foundational models to simulate operations, gain insights, and make informed decisions for effective planning and optimization.

By relying on various models, Bedrock gains insights, predicts outcomes, and makes informed decisions. It continuously refines its models with real-world data, allowing it to adapt and optimize its operations.

Amazon Bedrock then plans for different scenarios and selects optimal strategies through simulations and adjustments to model parameters.

### Advantages of model-based reflex agents

- Quick and efficient decision-making based on their understanding of the world
- Better equipped to make accurate decisions by constructing an internal model of the world
- Adaptability to changes in the environment by updating their internal models
- More informed and strategic choices by using its internal state and rules to determine the condition

### Disadvantages of model-based reflex agents

- Building and maintaining models can be computationally expensive

- The models may not capture the real-world environment's complexity very well
- Models cannot anticipate all potential situations that may arise
- Models need to be updated often to stay current
- Models may pose challenges in terms of interpretation and comprehension

### 3. Goal-based Agents

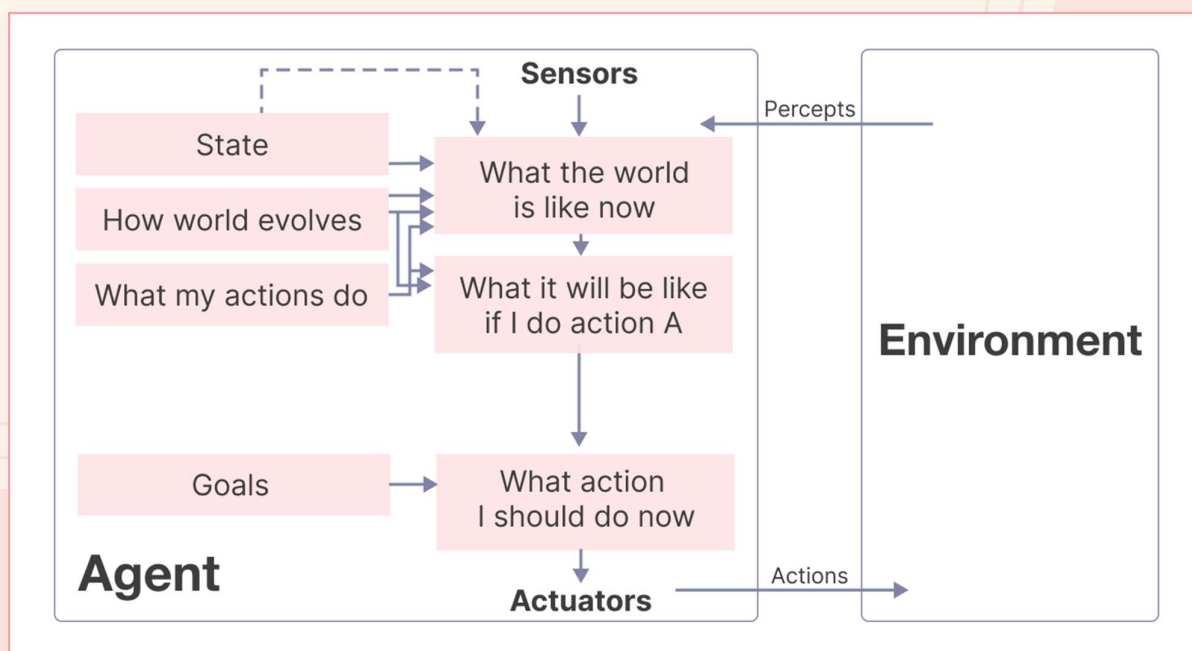
Goal-based agents are AI agents that use information from their environment to achieve specific goals. They employ search algorithms to find the most efficient path towards their objectives within a given environment.

These agents are also known as rule-based agents, as they follow predefined rules to accomplish their goals and take specific actions based on certain conditions.

Goal-based agents are easy to design and can handle complex tasks. They can be used in various applications like robotics, computer vision, and natural language processing.

Unlike basic models, a goal-based agent can determine the optimal course of decision-making and action-taking processes depending on its desired outcome or goal.

## Goal-based Agent





## How does it work?

Given a plan, a goal-based agent attempts to choose the best strategy to achieve the goals, it then uses search algorithms and heuristics to find the efficient path to the goal.

The working pattern of the goal-based agent can be divided into five steps:

1. **Perception:** The agent perceives its environment using sensors or other input devices to collect information about its surroundings.
2. **Reasoning:** The agent analyses the information collected and decides on the best course of action to achieve its goal.
3. **Action:** The agent takes actions to achieve its goal, such as moving or manipulating objects in the environment.
4. **Evaluation:** After taking action, the agent evaluates its progress towards the goal and adjusts its actions, if necessary.
5. **Goal Completion:** Once the agent has achieved its goal, it either stops working or begins working on a new goal.

## Example

We can say that [Google Bard](#) is a goal-based agent. No doubt, it is also a learning agent.

As a goal-based agent, it has a goal or objective to provide high-quality responses to user queries. It chooses its actions that are likely to assist users in finding the information they seek and achieving their desired goal of obtaining accurate and helpful responses.

## Advantages of goal-based agents

- Simple to implement and understand
- Efficient for achieving a specific goal
- Easy to evaluate performance based on goal completion
- It can be combined with other AI techniques to create more advanced agents

- Well-suited for well-defined, structured environments
- It can be used for various applications, such as robotics, game AI, and autonomous vehicles.

### **Disadvantages of goal-based agents**

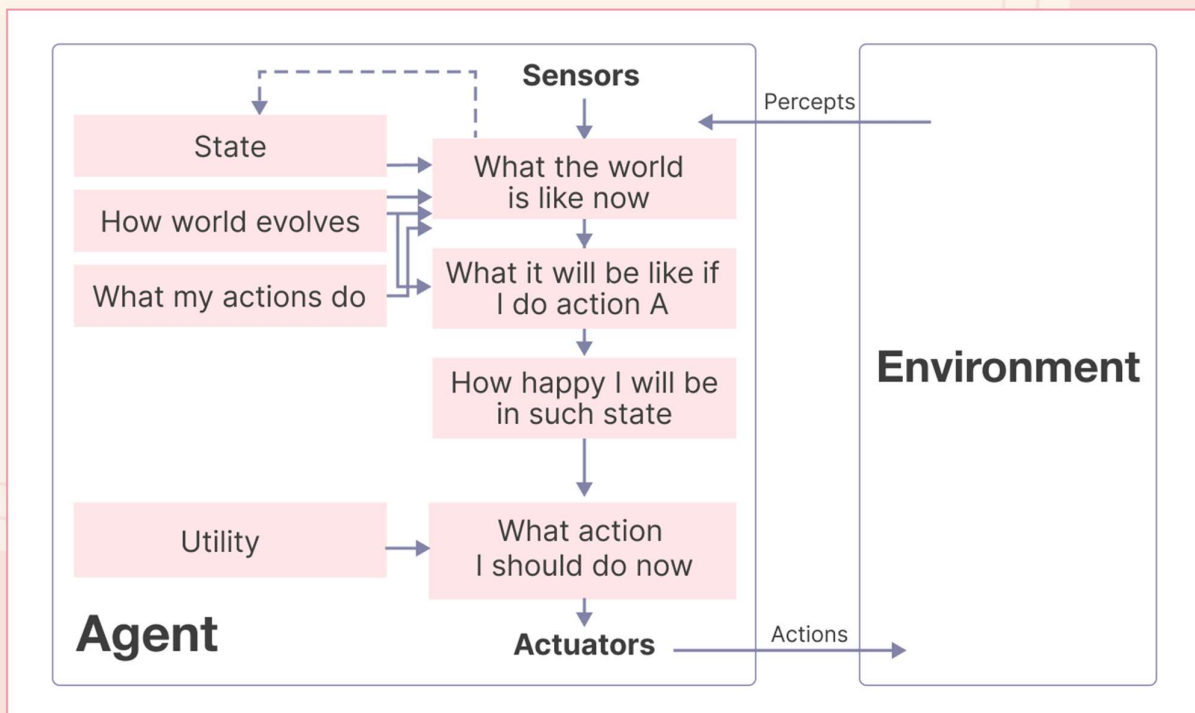
- Limited to a specific goal
- Unable to adapt to changing environments
- Ineffective for complex tasks that have too many variables
- Requires significant domain knowledge to define goals

### **4. Utility-based Agents**

Utility-based agents are AI agents that make decisions based on maximizing a utility function or value. They choose the action with the highest expected utility, which measures how good the outcome is.

This helps them deal with complex and uncertain situations more flexibly and adaptively. Utility-based agents are often used in applications where they have to compare and select among multiple options, such as resource allocation, scheduling, and game-playing.

# Utility-based Agent



 SIMFORM

## How does it work?

A utility-based agent aims to choose actions that lead to a high utility state. To achieve this, it needs to model its environment, which can be simple or complex.

Then, it evaluates the expected utility of each possible outcome based on the probability distribution and the utility function.

Finally, it selects the action with the highest expected utility and repeats this process at each time step.

## Example

[Anthropic Claude](#), an AI tool whose goal is to help cardmembers maximize their rewards and benefits from using cards, is a utility-based agent.

Because to achieve its goal, it employs a utility function to assign numerical values representing success or happiness to different states (situations that cardmembers face, such as purchasing, paying bills, redeeming rewards, etc.). And then compares the outcome of different actions in each state and trade-off decisions based on their utility values.

Furthermore, it uses heuristics and AI techniques to simplify and improve decision-making.

### **Advantages of utility-based agents**

- Handles a wide range of decision-making problems
- Learns from experience and adjusts their decision-making strategies
- Offers a consistent and objective framework for decision-making

### **Disadvantages of utility-based agents**

- Requires an accurate model of the environment, failing to do so results in decision-making errors
- Computationally expensive and requires extensive calculations
- Does not consider moral or ethical considerations
- Difficult for humans to understand and validate

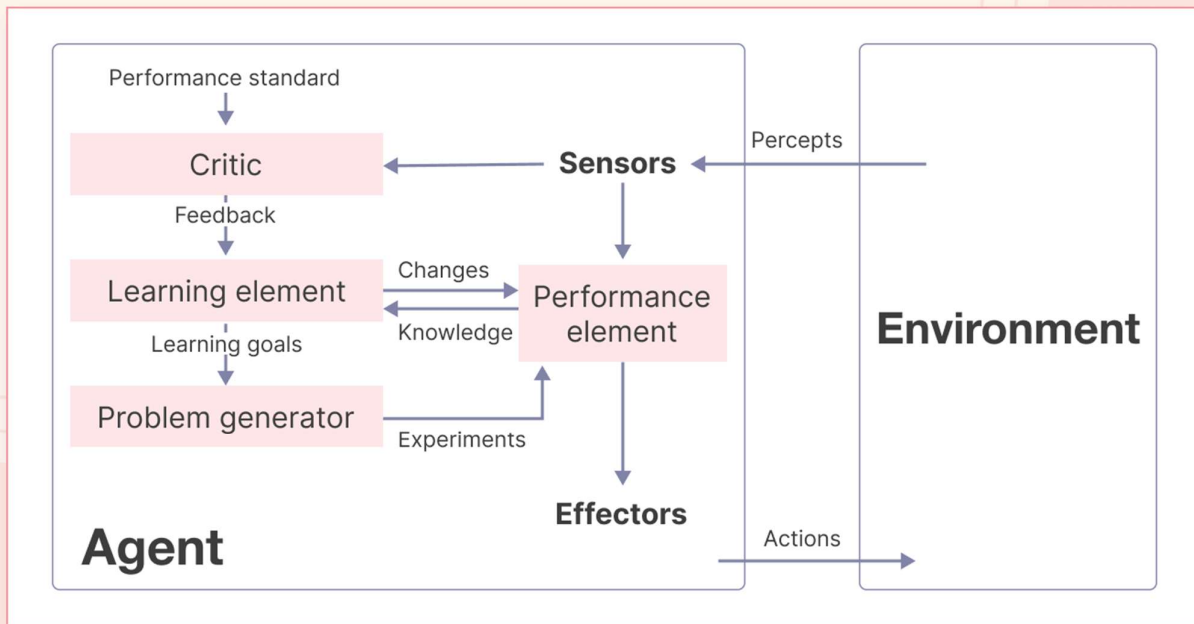
## **5. Learning Agents**

An AI learning agent is a software agent that can learn from past experiences and improve its performance. It initially acts with basic knowledge and adapts automatically through machine learning.

The learning agent comprises four main components:

- **Learning Element:** It is responsible for learning and making improvements based on the experiences it gains from its environment.
- **Critic:** It provides feedback to the learning element by the agent's performance for a predefined standard.
- **Performance Element:** It selects and executes external actions based on the information from the learning element and the critic.
- **Problem Generator:** It suggests actions to create new and informative experiences for the learning element to improve its performance.

# Learning Agent



## How does it work?

AI learning agents follow a cycle of observing, learning, and acting based on feedback. They interact with their environment, learn from feedback, and modify their behaviour for future interactions.

Here's how the cycle works:

- **Observation:** The learning agent observes its environment through sensors or other inputs.
- **Learning:** The agent analyses data using algorithms and statistical models, learning from feedback on its actions and performance.
- **Action:** Based on what it has learned, the agent acts in its environment to decide how to behave.
- **Feedback:** The agent receives feedback about their actions and performance through rewards, penalties, or environmental cues.
- **Adaptation:** Using feedback, the agent changes its behavior and decision-making processes, updating its knowledge and adapting to its environment.

This cycle repeats over time, allowing the agent to continuously improve its performance and adapt to changing circumstances.

### **Example**

A good example of a learning agent program is AutoGPT, created by Significant Gravititas.

Imagine you want to purchase a smartphone. So, you give [AutoGPT](#) a prompt to conduct market research on the top ten smartphones, providing insights on their pros and cons.

Once given this task, AutoGPT analyses the pros and cons of the top ten smartphones by exploring various websites and sources. It evaluates the authenticity of websites using a sub-agent program. Finally, it generates a detailed report summarizing the findings and listing the pros and cons of the top ten smartphone companies.

### **Advantages of learning agents**

- The agent can convert ideas into action based on AI decisions
- Learning intelligent agents can follow basic commands, like spoken instructions, to perform tasks
- Unlike classic agents that perform predefined actions, learning agents can evolve with time
- AI agents consider utility measurements, making them more realistic

### **Disadvantages of learning agents**

- Prone to biased or incorrect decision-making
- High development and maintenance costs
- Requires significant computing resources
- Dependence on large amounts of data
- Lack of human-like intuition and creativity

## How does an AI agent work?

An AI agent works by perceiving its environment through sensors, processing this information to understand the current state, making decisions based on predefined goals or objectives, and taking actions using effectors to achieve those goals. This process involves using algorithms, models, and data to learn, reason, and adapt over time, allowing the agent to interact autonomously and intelligently with its environment.

<https://yellow.ai/blog/ai-agents/#:~:text=at%20different%20levels,-.How%20does%20an%20AI%20agent%20work%3F%20A%20detailed%20perspective,-An%20AI%20agent%E2%80%99s>

## The Structure of Agents in Artificial Intelligence

Agents in Artificial Intelligence follow this simple structural formula:

Architecture + Agent Program = Agent

These are the terms most associated with agent structure:

- **Architecture:** This is the machinery or platform that executes the agent. It is a device with sensors and actuators, for example, a robotic car.
- **Agent Function:** The agent function maps a precept to the Action, represented by the following formula:  $f:P^* - A$
- **Agent Program:** The agent program is an implementation of the agent function. The agent program produces function  $f$  by executing on the physical architecture.

Many AI Agents use the PEAS model in their structure. PEAS is an acronym for Performance Measure, Environment, Actuators, and Sensors.

For instance, take a vacuum cleaner.

Performance: Cleanliness and efficiency

Environment: Rug, hardwood floor, living room

Actuator: Brushes, wheels, vacuum bag

Sensors: Dirt detection sensor, bump sensor

The structure of an AI agent typically consists of several components:

- **Perception:** The agent receives input from its environment or sensors. This includes data about the current state of the world or problem.
- **Knowledge Base:** The agent maintains a knowledge base or memory to store information and past experiences, which it can reference when making decisions.
- **Reasoning/Inference:** The agent uses its knowledge and reasoning mechanisms to draw conclusions, make predictions, or derive solutions based on the available information.
- **Decision-Making:** The agent's decision-making component selects actions or plans based on the results of reasoning. It may involve algorithms, heuristics, or optimization techniques.
- **Actuators:** The agent takes actions in the environment using actuators or effectors. These actions can impact the environment or help the agent achieve its goals.
- **Goal/Task Specification:** The agent is guided by goals, objectives, or tasks that define what it aims to achieve. These goals may come from external sources or be internally defined.
- **Learning Component:** Some agents can learn and adapt from experience. They may employ machine learning algorithms or other techniques to improve their performance.
- **Communication:** In multi-agent systems, agents may communicate with each other to exchange information or coordinate their actions.

The specific structure and components of an AI agent can vary widely depending on the application, problem, and the type of agent (e.g., expert systems, robotic agents,



software agents). Agents can range from simple rule-based systems to highly complex machine learning models.

### **What is a rational agent with an example?**

A rational agent is a computer program that uses logical reasoning and the ability to make decisions to determine its following action. It is a computer program that performs tasks based on pre-defined rules and procedures. The idea is that the agent can be programmed to follow specific instructions to make decisions rather than requiring its programmer to write every decision down manually.

An excellent example of a rational agent is a chess player. A chess player can analyze the board and determine which moves will result in the most advantageous outcome for itself. It can move one piece over another, move its king out of danger, or attack an opponent's piece.

### **What are the four characteristics of a rational agent?**

A rational agent has four primary characteristics:

- **Perception:** The ability to perceive the current state of the environment and gather relevant information.
- **Actuators:** The ability to take actions within the environment to achieve its goals.
- **Performance measure:** A way to evaluate the success or failure of the agent's actions.
- **Rationality:** The ability to make decisions based on logical reasoning and optimize behaviour to achieve its goals, considering its perception of the environment and the performance measure.

### **What do you mean by Table Driven Agent? What are the disadvantages or problems of this agent?**

A Table-Driven Agent is an approach in artificial intelligence where the agent's behaviour is determined by a predefined lookup table or set of rules. Each entry in

the table specifies the action the agent should take based on the current percept or state of the environment.

Disadvantages or problems of a Table-Driven Agent include:

1. **Limited Flexibility:** Table-driven agents are rigid and lack the ability to adapt to new or changing environments. They can only respond to situations predefined in the lookup table.
2. **Scalability Issues:** As the complexity of the environment or the number of possible states/actions increases, maintaining and updating the lookup table becomes cumbersome and impractical.
3. **Inefficiency:** Table-driven agents may require large amounts of memory to store the lookup table, especially for environments with many possible states or actions.
4. **Difficulty in Encoding Knowledge:** Designing an effective lookup table requires prior knowledge of the environment and all possible interactions, which may not always be feasible or accurate.
5. **Limited Learning Capability:** Table-driven agents cannot learn from experience or improve their performance over time since their behaviour is solely determined by the predefined table of rules.

**Explain Properties of environment with examples.**

**Fully observable vs partially observable environment**

In types of environments in AI, the first one can be classified as fully observable or partially observable, depending on the extent to which the agent has access to information about the current state of the environment.

- A fully observable environment is one in which the agent has complete information about the current state of the environment. The agent has direct access to all environmental features that are necessary for making decisions. Examples of fully observable environments include board games like chess or checkers.
- A partially observable environment is one in which the agent does not have complete information about the current state of the environment. The agent can only observe a subset of the environment, and some aspects of the

environment may be hidden or uncertain. Examples of partially observable environments include driving a car in traffic.

### **Deterministic vs Stochastic**

The environment in artificial intelligence can be classified as deterministic or stochastic, depending on the level of predictability of the outcomes of the agent's actions.

- A deterministic environment is one in which the outcome of an action is completely predictable and can be precisely determined. The state of the environment completely determines the result of an agent's action. In a deterministic environment, the agent's actions have a one-to-one correspondence with the resulting outcomes. Examples of deterministic environments include simple mathematical equations, where the outcome of each operation is precisely defined.
- A stochastic environment is one in which the outcome of an action is uncertain and involves probability. The state of the environment only partially determines the result of an agent's action, and there is a degree of randomness or unpredictability in the outcome. Examples of stochastic environments include games of chance like poker or roulette, where the outcome of each action is influenced by random factors like the shuffle of cards or the spin of a wheel.

### **Competitive vs Collaborative**

In types of environment in AI, another one can be classified as competitive or collaborative, depending on whether the agents in it are competing against each other or working together to achieve a common goal.

- A competitive environment is one in which multiple agents are trying to achieve conflicting goals. Each agent's success is directly tied to the failure of others, and the agents must compete against each other to achieve their objectives. Examples of competitive environments include games like chess.
- A collaborative environment is one in which multiple agents are working together to achieve a common goal. The success of each agent is directly tied to the success of the group as a whole, and the agents must collaborate and coordinate their actions to achieve their objectives. Examples of collaborative environments include tasks like search and rescue.

## Single-agent vs Multi-agent

The environment in Artificial Intelligence can be classified as a single-agent or multi-agent environment, depending on the number of agents interacting within it.

- A single-agent environment is one in which a single agent interacts with the environment to achieve its goals. Examples of single-agent environments include puzzles and mazes. The agent must use search algorithms or planning techniques to find a path to its goal state.
- A multi-agent environment is one in which multiple agents interact with each other and the environment to achieve their individual or collective goals. Examples of multi-agent environments include multiplayer games and traffic simulations. The agents must use game theory or multi-agent reinforcement learning techniques to optimize their behavior.

## Static vs Dynamic

Types of environment in AI can be classified as static or dynamic, depending on whether it changes over time.

- A static environment is one in which the environment does not change over time. The state of the environment remains constant, and the agent's actions do not affect the environment. Examples of static environments include mathematical problems or logic puzzles. The agent can use techniques like search algorithms or decision trees to optimize its behavior.
- A dynamic environment is one in which the environment changes over time. The state of the environment evolves based on the actions of the agent and other factors, and the agent's actions can affect the future state of the environment. Examples of dynamic environments include video games or robotics applications. The agent must use techniques like planning or reinforcement learning to optimize its behavior in response to the changing environment.

## Discrete vs Continuous

The environment in Artificial Intelligence can be classified as discrete or continuous, depending on the nature of the state and action spaces.

- The state space refers to the set of all possible states that the environment can be in. For example, in a game of chess, the state space would include all possible board configurations. In a robotic control task, the state space may

include information about the position and velocity of the robot and its environment.

- The action space refers to the set of all possible actions that the agent can take in each state of the environment. For example, in a game of chess, the action space would include all possible moves that the player can make. In a robotic control task, the action space may include commands for controlling the speed and direction of the robot.
- A discrete environment is one in which the state and action spaces are finite and discrete. Examples of discrete environments include board games like chess or checkers. The agent's decision-making process can be based on techniques like search algorithms or decision trees.
- In contrast, a continuous environment is one in which the state and action spaces are continuous and infinite. Examples of continuous environments include robotics or control systems. In a continuous environment, the agent's decision-making process must take into account the continuous nature of the state and action spaces. The agent must use techniques like reinforcement learning or optimization to learn and optimize its behavior.

### **Episodic vs Sequential**

In AI, an environment can be classified as episodic or sequential, depending on the nature of the task and the relationship between the agent's actions and the environment.

- An episodic environment is one in which the agent's actions do not affect the future states of the environment. The goal of the agent is to maximize the immediate reward obtained during each episode. Examples of episodic environments include games like chess. The agent can use techniques like Monte Carlo methods or Q-learning to learn the optimal policy for each episode.
- In contrast, a sequential environment is one in which the agent's actions affect the future states of the environment. The goal of the agent is to maximize the cumulative reward obtained over multiple interactions. Examples of sequential environments include robotics applications or video games. The agent must use techniques like dynamic programming or reinforcement learning to learn the optimal policy over multiple interactions.

## Known vs Unknown

The environment in Artificial Intelligence can be classified as a known or unknown environment, depending on the agent's level of knowledge about the environment.

- A known environment is one in which the agent has complete knowledge of the environment's rules, state transitions, and reward structure. The agent knows exactly what actions are available to it, and the outcome of each action is known with certainty. Examples of known environments include chess or tic-tac-toe games. In a known environment, the agent can use techniques like search algorithms or decision trees to optimize its behavior.
- In contrast, an unknown environment is one in which the agent has limited or no knowledge about the environment's rules, state transitions, and reward structure. The agent may not know what actions are available to it, or the outcome of each action may be uncertain. Examples of unknown environments include exploration tasks or real-world applications. In an unknown environment, the agent must use techniques like reinforcement learning or exploration-exploitation trade-offs to optimize its behavior.
- It's important to note that Known vs Unknown and Fully observable vs partially observable environments are independent of each other. For example, an environment could be known and partially observable, or unknown and fully observable. The choice of which characterization to use depends on the specific problem being addressed and the capabilities of the agent.

### List down all the characteristics of an intelligent agent.

1. **Autonomy:** Ability to operate independently and make decisions without human intervention.
2. **Perception:** Capability to sense and perceive the environment through sensors or observation mechanisms.
3. **Reasoning:** Capacity to analyze information, infer relationships, and make decisions based on logical or probabilistic reasoning.
4. **Learning:** Ability to acquire knowledge and improve performance through experience, feedback, or data.
5. **Adaptability:** Capability to adjust behavior and strategies in response to changes in the environment or goals.

6. **Goal-directedness:** Pursuit of predefined goals or objectives, guiding actions and decision-making.
7. **Communication:** Capacity to interact with other agents or entities, exchanging information or coordinating actions.
8. **Flexibility:** Capacity to handle uncertainty, ambiguity, and variability in the environment or tasks.
9. **Proactiveness:** Ability to anticipate future events or outcomes and take proactive actions to achieve goals.
10. **Social intelligence:** Capability to understand and navigate social interactions and dynamics, when applicable.

## All About Problem-Solving Agents in Artificial Intelligence

Problem-solving Agents in Artificial Intelligence employ several algorithms and analyses to develop solutions.

They are:

- **Search Algorithms:** Search techniques are considered universal problem-solving methods. Problem-solving or rational agents employ these algorithms and strategies to solve problems and generate the best results.

**Uninformed Search Algorithms:** Also called a Blind search, uninformed searches have no domain knowledge, working instead in a brute-force manner.

**Informed Search Algorithms:** Also known as a Heuristic search, informed searches use domain knowledge to find the search strategies needed to solve the problem.

- **Hill Climbing Algorithms:** Hill climbing algorithms are local search algorithms that continuously move upwards, increasing their value or elevation until they find the best solution to the problem or the mountain's peak.

Hill climbing algorithms are excellent for optimizing mathematical problem-solving. This algorithm is also known as a "greedy local search" because it only checks out its good immediate neighbour.

- **Means-Ends Analysis:** The means-end analysis is a problem-solving technique used to limit searches in Artificial Intelligence programs, combining Backward and Forward search techniques.

The means-end analysis evaluates the differences between the Initial State and the Final State, then picks the best operators that can be used for each difference. The analysis then applies the operators to each matching difference, reducing the current and goal state difference.

## **Problem-solving Agents**

Building agents with rational behaviour is the subject of artificial intelligence research. These agents often use a search algorithm in the background to complete their job. Search techniques are universal problem-solving approaches in Artificial Intelligence. Rational or problem-solving agents mostly use these search strategies or algorithms in AI to solve a particular problem and provide the best result. The goal-based agents are problem-solving agents that use atomic representation. We will learn about different problem-solving search algorithms in AI in this topic.

## **Search Algorithm Terminologies**

1. **Search** - Searching solves a search issue in a given space step by step. Three major factors can influence a search issue.
  - Search Space - A search space is a collection of potential solutions a system may have.
  - Start State - The jurisdiction where the agent starts the search.
  - Goal test - A function that examines the current state and returns whether or not the goal state has been attained.
2. **Search tree** - A Search tree is a tree representation of a search issue. The node at the root of the search tree corresponds to the initial condition.
3. **Actions** - It describes all the steps, activities, or operations accessible to the agent.
4. **Transition model** - It can be used to convey a description of what each action does.
5. **Path Cost** - It is a function that gives a cost to each path.



6. **Solution** - An action sequence connects the start node to the target node.
7. **Optimal Solution** - If a solution has the lowest cost among all solutions, it is said to be the optimal answer.

### Properties of Search Algorithms

The four important properties of search algorithms in artificial intelligence for comparing their efficiency are as follows:

**Completeness** - A search algorithm is said to be complete if it guarantees to yield a solution for any random input if at least one solution exists.

**Optimality** - A solution discovered for an algorithm is considered optimal if it is assumed to be the best solution (lowest path cost) among all other solutions.

**Time complexity** - It measures how long an algorithm takes to complete its job.

**Space Complexity** - The maximum storage space required during the search, as determined by the problem's complexity.

These characteristics often contrast the effectiveness of various search algorithms in artificial intelligence.

### Importance of Search Algorithms in Artificial Intelligence

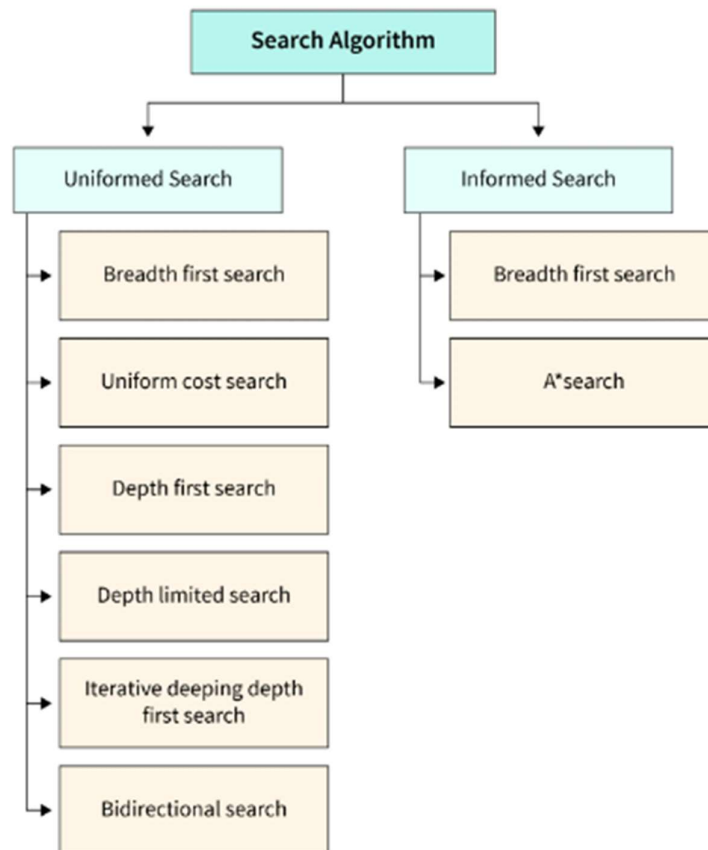
The following points explain how and why the search algorithms in AI are important:

- **Solving problems:** Using logical search mechanisms, including problem description, actions, and search space, search algorithms in artificial intelligence improve problem-solving. Applications for route planning, like Google Maps, are one real-world illustration of how search algorithms in AI are utilized to solve problems. These programs employ search algorithms to determine the quickest or shortest path between two locations.
- **Search programming:** Many AI activities can be coded in terms of searching, which improves the formulation of a given problem's solution.
- **Goal-based agents:** Goal-based agents' efficiency is improved through search algorithms in artificial intelligence. These agents look for the most optimal course of action that can offer the finest resolution to an issue to solve it.

- **Support production systems:** Search algorithms in artificial intelligence help production systems run. These systems help AI applications by using rules and methods for putting them into practice. Production systems use search algorithms in artificial intelligence to find the rules that can lead to the required action.
- **Neural network systems:** The neural network systems also use these algorithms. These computing systems comprise a hidden layer, an input layer, an output layer, and coupled nodes. Neural networks are used to execute many tasks in artificial intelligence. For example, the search for connection weights that will result in the required input-output mapping is improved by search algorithms in AI.

### **Types of Search Algorithms in AI**

We can divide search algorithms in artificial intelligence into uninformed (Blind search) and informed (Heuristic search) algorithms based on the search issues.



## Uninformed/Blind Search

The uninformed search needs domain information, such as proximity or goal location. It works by brute force because it only contains information on traversing the tree and identifying leaf and goal nodes.

Uninformed search is a method of searching a search tree without knowledge of the search space, such as initial state operators and tests for the objective, and is also known as blind search. It goes through each tree node until it reaches the target node. These algorithms are limited to producing successors and distinguishing between goal and non-goal states.

### 1. Breadth-first Search:

- Breadth-first search is the most common search strategy for traversing a tree or graph. This algorithm searches breadthwise in a tree or graph, so it is called breadth-first search.
- BFS algorithm starts searching from the root node of the tree and expands all successor node at the current level before moving to nodes of next level.
- The breadth-first search algorithm is an example of a general-graph search algorithm.
- Breadth-first search implemented using FIFO queue data structure.

### Advantages:

- BFS will provide a solution if any solution exists.
- If there are more than one solution for a given problem, then BFS will provide the minimal solution which requires the least number of steps.

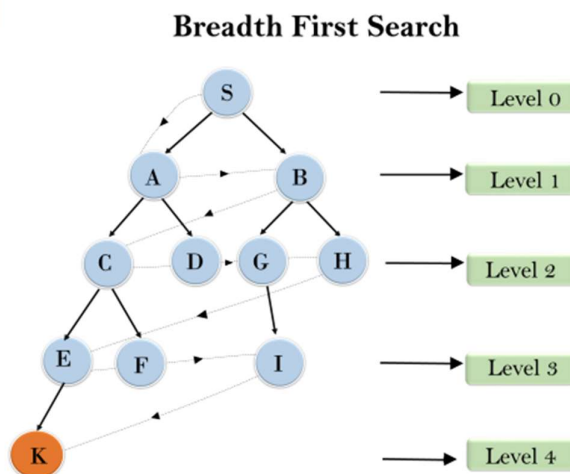
### Disadvantages:

- It requires lots of memory since each level of the tree must be saved into memory to expand the next level.
- BFS needs lots of time if the solution is far away from the root node.

Example:

In the below tree structure, we have shown the traversing of the tree using BFS algorithm from the root node S to goal node K. BFS search algorithm traverse in layers, so it will follow the path which is shown by the dotted arrow, and the traversed path will be:

S----> A---->B----->C---->D----->G---->H--->E----->F----->I----->K



**Time Complexity:** Time Complexity of BFS algorithm can be obtained by the number of nodes traversed in BFS until the shallowest Node. Where the  $d$  = depth of shallowest solution and  $b$  is a node at every state.

$$T(b) = 1 + b^2 + b^3 + \dots + b^d = O(b^d)$$

**Space Complexity:** Space complexity of BFS algorithm is given by the Memory size of frontier which is  $O(b^d)$ .

**Completeness:** BFS is complete, which means if the shallowest goal node is at some finite depth, then BFS will find a solution.

**Optimality:** BFS is optimal if path cost is a non-decreasing function of the depth of the node.

## 2. Depth-first Search

- Depth-first search is a recursive algorithm for traversing a tree or graph data structure.
- It is called the depth-first search because it starts from the root node and follows each path to its greatest depth node before moving to the next path.
- DFS uses a stack data structure for its implementation.
- The process of the DFS algorithm is similar to the BFS algorithm.

### Advantage:

- DFS requires very less memory as it only needs to store a stack of the nodes on the path from root node to the current node.
- It takes less time to reach to the goal node than BFS algorithm (if it traverses in the right path).

### Disadvantage:

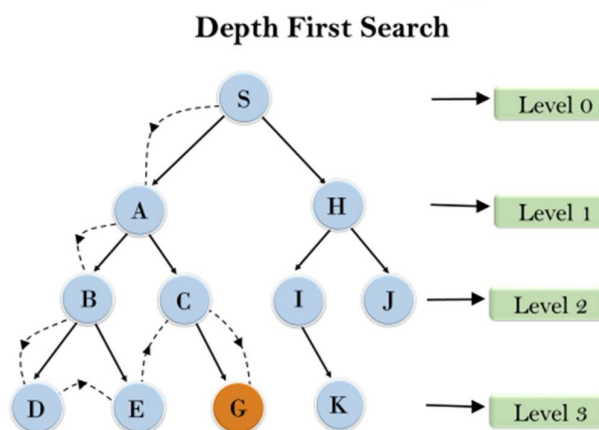
- There is the possibility that many states keep re-occurring, and there is no guarantee of finding the solution.
- DFS algorithm goes for deep down searching and sometime it may go to the infinite loop.

### Example:

In the below search tree, we have shown the flow of depth-first search, and it will follow the order as:

Root node--->Left node ----> right node.

It will start searching from root node S, and traverse A, then B, then D and E, after traversing E, it will backtrack the tree as E has no other successor and still goal node is not found. After backtracking it will traverse node C and then G, and here it will terminate as it found goal node.



**Completeness:** DFS search algorithm is complete within finite state space as it will expand every node within a limited search tree.

**Time Complexity:** Time complexity of DFS will be equivalent to the node traversed by the algorithm. It is given by:

$$T(n) = 1 + n^2 + n^3 + \dots + n^m = O(n^m)$$

**Where, m = maximum depth of any node and this can be much larger than d (Shallowest solution depth)**

**Space Complexity:** DFS algorithm needs to store only single path from the root node, hence space complexity of DFS is equivalent to the size of the fringe set, which is **O(bm)**.

**Optimal:** DFS search algorithm is non-optimal, as it may generate a large number of steps or high cost to reach to the goal node.

### 3. Depth-Limited Search Algorithm:

A depth-limited search algorithm is similar to depth-first search with a predetermined limit. Depth-limited search can solve the drawback of the infinite path in the Depth-first search. In this algorithm, the node at the depth limit will treat as it has no successor nodes further.

#### Depth-limited search can be terminated with two Conditions of failure:

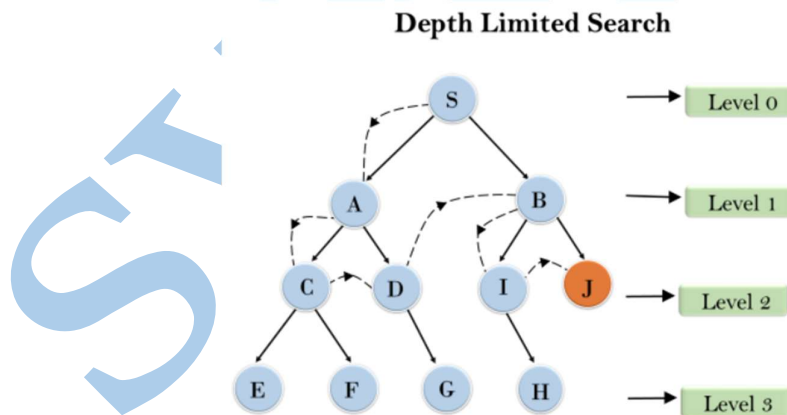
- **Standard failure value:** It indicates that problem does not have any solution.
- **Cutoff failure value:** It defines no solution for the problem within a given depth limit.

#### Advantages:

Depth-limited search is Memory efficient.

#### Disadvantages:

- Depth-limited search also has a disadvantage of incompleteness.
- It may not be optimal if the problem has more than one solution.



**Completeness:** DLS search algorithm is complete if the solution is above the depth-limit.

**Time Complexity:** Time complexity of DLS algorithm is  $O(b^l)$ .

**Space Complexity:** Space complexity of DLS algorithm is  $O(b \times l)$ .

**Optimal:** Depth-limited search can be viewed as a special case of DFS, and it is also not optimal even if  $l > d$ .

#### 4. Uniform-cost Search Algorithm:

Uniform-cost search is a searching algorithm used for traversing a weighted tree or graph. This algorithm comes into play when a different cost is available for each edge. The primary goal of the uniform-cost search is to find a path to the goal node which has the lowest cumulative cost. Uniform-cost search expands nodes according to their path costs from the root node. It can be used to solve any graph/tree where the optimal cost is in demand. A uniform-cost search algorithm is implemented by the priority queue. It gives maximum priority to the lowest cumulative cost. Uniform cost search is equivalent to BFS algorithm if the path cost of all edges is the same.

##### Advantages:

- Uniform cost search is optimal because at every state the path with the least cost is chosen.

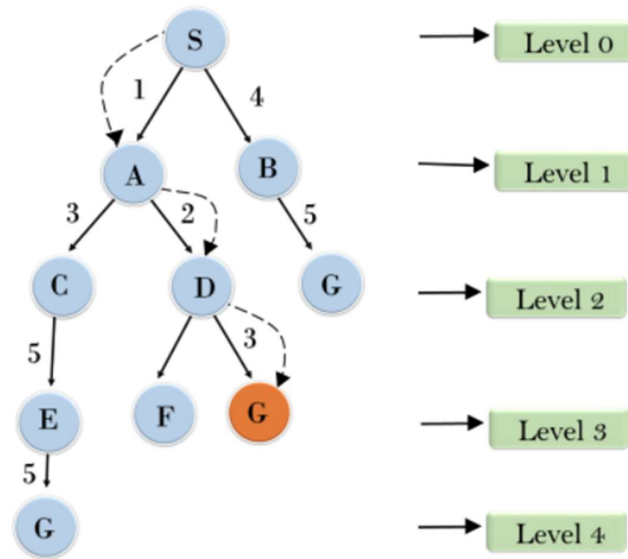
##### Disadvantages:

- It does not care about the number of steps involve in searching and only concerned about path cost. Due to which this algorithm may be stuck in an infinite loop.

##### Example:



## Uniform Cost Search



### Completeness:

Uniform-cost search is complete, such as if there is a solution, UCS will find it.

### Time Complexity:

Let  $C^*$  is **Cost of the optimal solution**, and  $\epsilon$  is each step to get closer to the goal node. Then the number of steps is  $= C^*/\epsilon + 1$ . Here we have taken  $+1$ , as we start from state 0 and end to  $C^*/\epsilon$ .

Hence, the worst-case time complexity of Uniform-cost search is  $O(b^{1 + \lceil C^*/\epsilon \rceil})$ .

### Space Complexity:

The same logic is for space complexity so, the worst-case space complexity of Uniform-cost search is  $O(b^{1 + \lceil C^*/\epsilon \rceil})$ .

**Optimal:** Uniform-cost search is always optimal as it only selects a path with the lowest path cost.

## 5. Iterative deepening depth-first Search:

The iterative deepening algorithm is a combination of DFS and BFS algorithms. This search algorithm finds out the best depth limit and does it by gradually increasing the limit until a goal is found.

This algorithm performs depth-first search up to a certain "depth limit", and it keeps increasing the depth limit after each iteration until the goal node is found.

This Search algorithm combines the benefits of Breadth-first search's fast search and depth-first search's memory efficiency.

The iterative search algorithm is useful uninformed search when search space is large, and depth of goal node is unknown.

**Advantages:**

- It combines the benefits of BFS and DFS search algorithm in terms of fast search and memory efficiency.

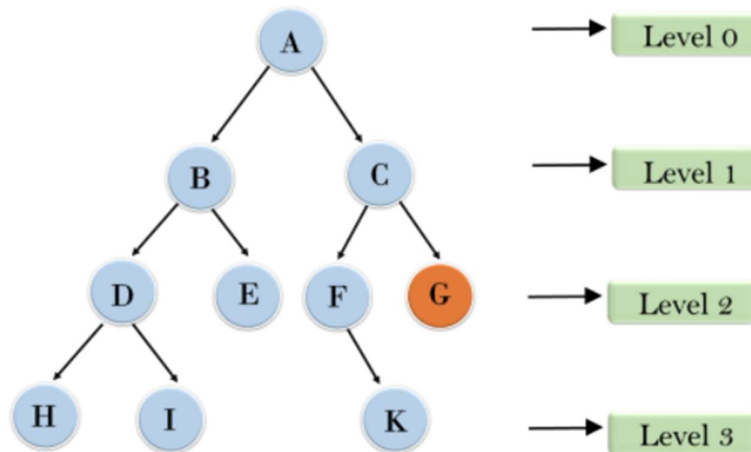
**Disadvantages:**

- The main drawback of IDDFS is that it repeats all the work of the previous phase.

**Example:**

Following tree structure is showing the iterative deepening depth-first search. IDDFS algorithm performs various iterations until it does not find the goal node. The iteration performed by the algorithm is given as:

## Iterative deepening depth first search



1'st Iteration-----> A

2'nd Iteration-----> A, B, C

3'rd Iteration----->A, B, D, E, C, F, G

4'th Iteration----->A, B, D, H, I, E, C, F, K, G

In the fourth iteration, the algorithm will find the goal node.

### Completeness:

This algorithm is complete is if the branching factor is finite.

### Time Complexity:

Let's suppose  $b$  is the branching factor and depth is  $d$  then the worst-case time complexity is  $O(b^d)$ .

### Space Complexity:

The space complexity of IDDFS will be  $O(bd)$ .

### Optimal:

IDDFS algorithm is optimal if path cost is a non- decreasing function of the depth of the node.

## 6. Bidirectional Search Algorithm:

Bidirectional search algorithm runs two simultaneous searches, one form initial state called as forward-search and other from goal node called as backward-search, to

find the goal node. Bidirectional search replaces one single search graph with two small subgraphs in which one starts the search from an initial vertex and other starts from goal vertex. The search stops when these two graphs intersect each other. Bidirectional search can use search techniques such as BFS, DFS, DLS, etc.

### Advantages:

- Bidirectional search is fast.
- Bidirectional search requires less memory

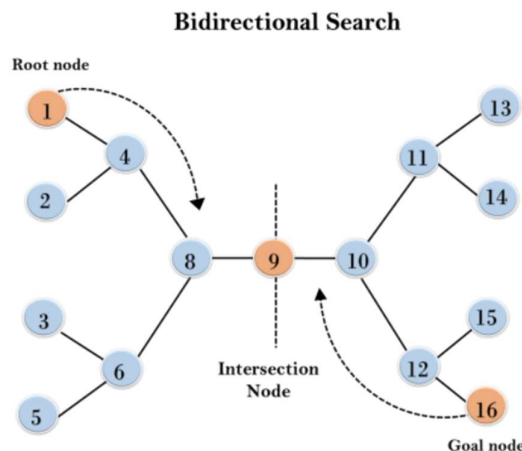
### Disadvantages:

- Implementation of the bidirectional search tree is difficult.
- **In bidirectional search, one should know the goal state in advance.**

### Example:

In the below search tree, bidirectional search algorithm is applied. This algorithm divides one graph/tree into two sub-graphs. It starts traversing from node 1 in the forward direction and starts from goal node 16 in the backward direction.

The algorithm terminates at node 9 where two searches meet.



**Completeness:** Bidirectional Search is complete if we use BFS in both searches.

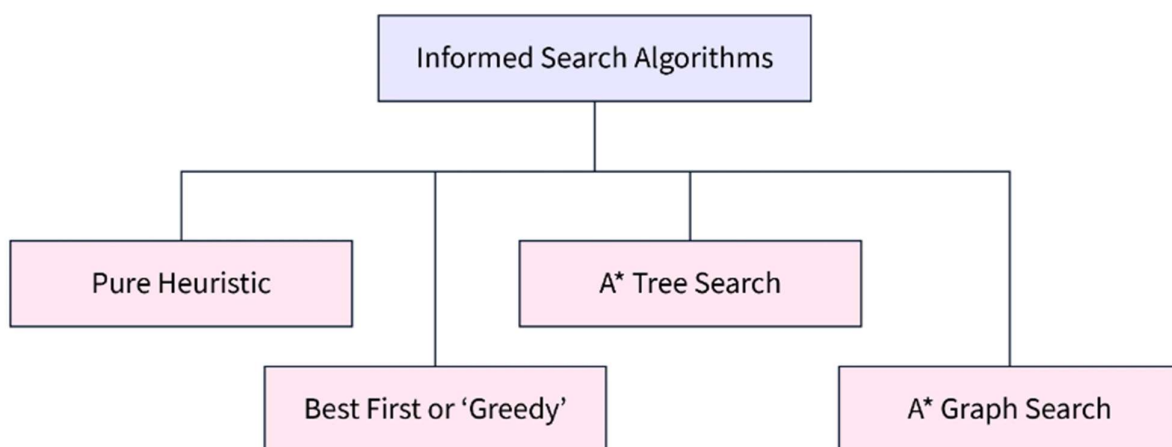
**Time Complexity:** Time complexity of bidirectional search using BFS is  $O(b^d)$ .

**Space Complexity:** Space complexity of bidirectional search is  $O(b^d)$ .

**Optimal:** Bidirectional search is Optimal.

### Informed search algorithms

Informed search algorithms are a type of search algorithm that uses heuristic functions to guide the search process. For example, a heuristic function calculates the cost of moving from a starting state to a goal state. By employing this assessment, informed search algorithms can select search paths more likely to lead to the desired state. As a result, the search process is improved, making it quicker and more accurate for AI systems to make decisions.



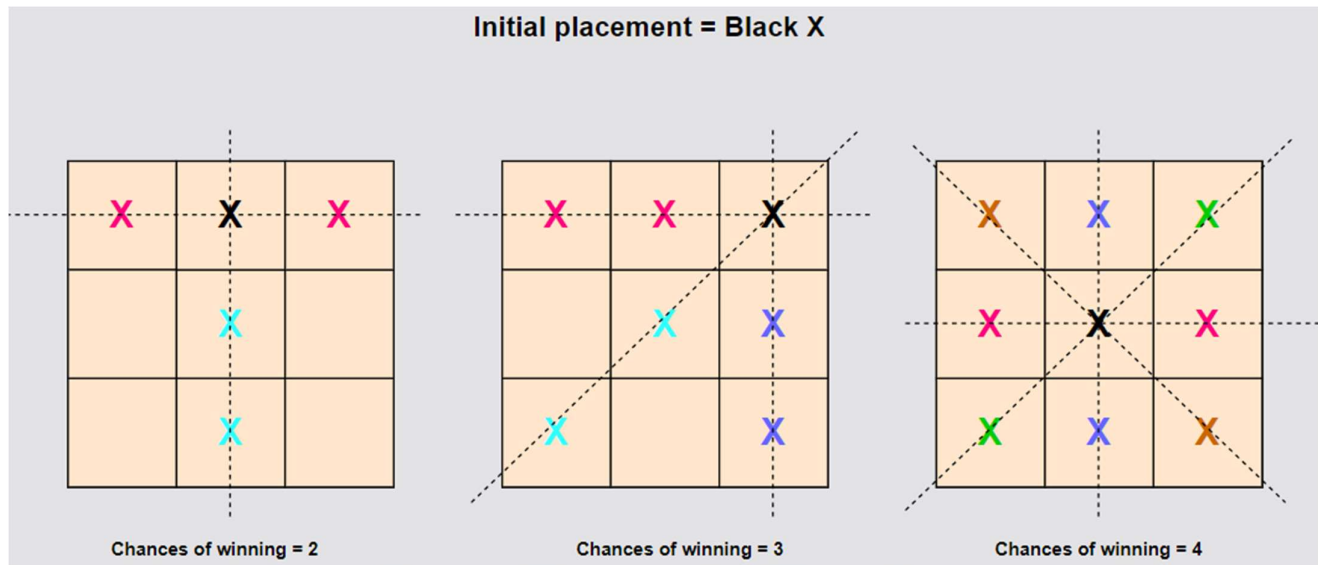
### Heuristic function

A **heuristic function** helps the search algorithm choose a branch from the ones that are available. It helps with the decision process by using some extra knowledge about the search space.

Let's use a simple analogy. If you went to a supermarket with many check-out counters, you would try to go to the one with the least number of people waiting. This is a heuristic that reduces your wait time.

## Example

While playing tic tac toe, there are many placements from which one player can start, and each placement has its own chances of winning. However, if the first player starts from the centremost area, they have the most chances of winning. Hence, *chances of winning* can be a heuristic.



## Examples of informed search algorithms

### 1) Best first search (Greedy Search)

The **best first search algorithm** is a version of the depth first search using heuristics. Each node is evaluated with respect to its distance from the goal. Whichever node is closest to the final state is explored first. If the path fails to reach the goal, the algorithm backtracks and chooses some other node that didn't seem to be the best before.

### Algorithm

1. Create a priority queue.
2. Insert the starting node.
3. Remove a node from the priority queue.
  - 3.1. Check if it is the goal node. If yes, then exit.

3.2. Otherwise, mark the node as visited and insert its neighbours into the priority queue. The priority of each will be its distance from the goal.

### **Advantages:**

- If a decent heuristic function is applied, it finds a solution rapidly.
- It can be applied to a variety of issues.
- Implementing it is simple.

### **Disadvantages:**

- As it only considers the expected distance to the target and not the actual cost of getting there, it could only sometimes come up with the best solution.
- It can become trapped in a local minimum and stop looking for alternate routes.
- To perform properly, it needs a good heuristic function.

## **2) A\* Search**

A\* uses the path of reaching to the current node from the starting node, and the path of reaching the goal from the current node. So, the heuristic function becomes:

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

where:

f(n): cost of the optimal path from start to goal

g(n): shortest path of the current node from the start

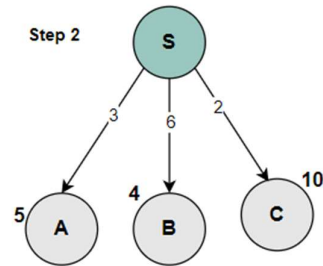
h(n): shortest path of the goal from the current node

Note: The actual distance from any node to the goal may be greater than h(n), since h(n) is the shortest distance. This distance can be the straight-line distance from the current node to the goal. A path with the shortest distance may or may not exist.

### **Algorithm**

1. Create a Priority Queue.
2. Insert the starting node.
3. Remove a node from the priority queue.
  - 3.1. Check if it is the goal node. If yes, then exit.

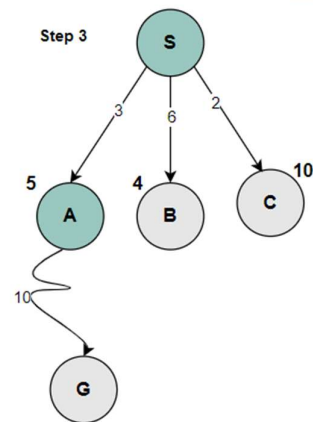
3.2. Otherwise, mark the node as visited and insert its neighbours into the priority queue. The priority of each node will be the sum of its cost from the start and the goal.



$$f(A) = 3 + 5 = 8$$

$$f(B) = 6 + 4 = 10$$

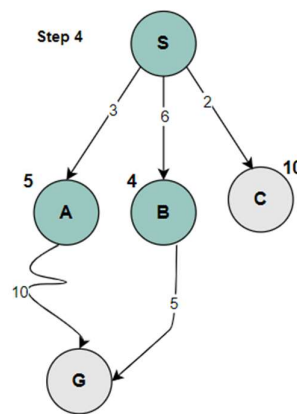
$$f(C) = 2 + 9 = 12$$



$$f(B) = 6 + 4 = 10$$

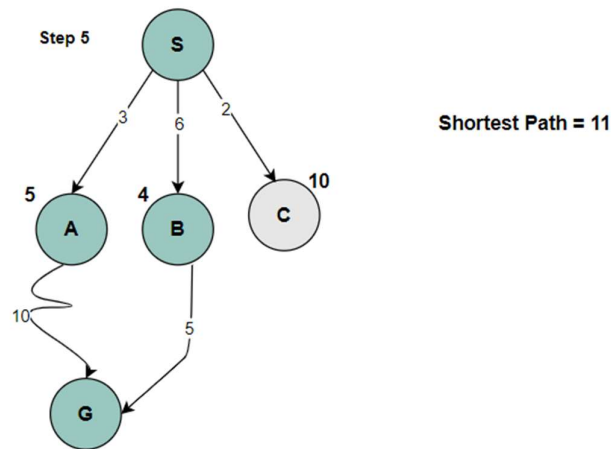
$$f(C) = 2 + 9 = 12$$

$$f(G) = g(G) = 13$$



Since the distance of G from S is shorter through B, it is updated  
 $f(G) = g(G) = 11$   
 $f(C) = 2 + 9 = 12$





### Advantages:

- If the heuristic function is acceptable, which means it never overestimates the real cost, the A\* Search Algorithm promises to identify the best course of action.
- Because it only investigates nodes with a good chance of leading to the target, the A\* Search Algorithm is effective in terms of time and space complexity.

### Disadvantages:

- If infinitely many nodes have a lower total cost than the goal node, the A\* Search Algorithm might not be complete.
- If the heuristic function is inadmissible or inconsistent, it overestimates or underestimates the actual cost. Therefore, there may be better options than the A\* Search Algorithm.

<https://www.scaler.com/topics/artificial-intelligence-tutorial/informed-search/>

### Difference between Informed and Uninformed Search

Feature	Informed Search	Uninformed Search
Knowledge Used	Utilizes additional knowledge about the problem domain (heuristics).	Operates solely on the problem structure without external knowledge.
Search Efficiency	Generally more efficient due to heuristic guidance.	Less efficient as it explores the search space systematically without guidance.
Heuristic Function	Employs a heuristic function to estimate the cost to the goal.	No heuristic function; treats each node uniformly.
Examples	A* Search, Greedy Best-First Search, IDA*.	Breadth-First Search (BFS), Depth-First Search (DFS), Uniform Cost Search.
Optimality	Can be optimal, depending on the heuristic used.	Typically optimal in finite, unweighted graphs (e.g., BFS) but not always (e.g., DFS).
Completeness	Complete in finite search spaces if the heuristic is admissible.	Complete in finite search spaces.

### Key Features of Informed Search Algorithms

- **Heuristic Function:** Informed search algorithms utilize a heuristic function to estimate the cost of the cheapest path from a given node to the goal. This function helps in prioritizing which nodes to explore next.
- **Efficiency:** By using domain-specific knowledge, informed searches are generally more efficient than uninformed ones, as they can often find solutions faster by exploring fewer nodes.
- **Optimality and Completeness:** Depending on the heuristic used, informed search algorithms can be both optimal (finding the best solution) and complete (guaranteeing a solution if one exists).
- **Examples of Informed Search Algorithms:** A\* search, Greedy Best-First Search, and Iterative Deepening A\* (IDA\*) are prominent examples of informed search algorithms.

### Key Features of Uninformed Search Algorithms

- **Lack of Heuristic Guidance:** Uninformed search strategies do not use a heuristic function. They rely solely on the problem structure and the rules of movement within the search space.
- **Uniformity:** These algorithms treat each node in the search space uniformly and explore them systematically, such as in breadth-first or depth-first order.
- **Completeness and Optimality:** Uninformed search algorithms are generally complete (they will find a solution if one exists), but they may not always be optimal (find the best solution). For example, Breadth-First Search (BFS) is complete and optimal for unweighted graphs, but Depth-First Search (DFS) is neither in infinite search spaces.
- **Examples of Uninformed Search Algorithms:** Common examples include Breadth-First Search (BFS), Depth-First Search (DFS), and Uniform Cost Search.

### **What do you mean by on-line search agents and unknown environments?**

Online search agents operate in unknown environments and must learn through interaction rather than pure computation. They interleave planning and action by first taking an action, observing the outcome, and using this information to plan their next action. These agents gather information and learn about the environment dynamically as they operate, adjusting their strategies based on the feedback they receive. Like hill-climbing search, online search agents keep only their current state in memory, acting locally to explore their environment through experimentation in real-time.

The canonical example of online search is a robot that is placed in a new building and must explore it to build a map that it can use for getting from A to B. Methods for escaping from labyrinths—required knowledge for aspiring heroes of antiquity—are also examples of online search algorithms. Spatial exploration is not the only form of exploration, however. Consider a newborn baby: it has many possible actions but knows the outcomes of none of them, and it has experienced only a few of the possible states that it can reach. The baby's gradual discovery of how the world works is, in part, an online search process.

Unknown environments refer to settings where the agent does not have prior knowledge of the layout, rules, or dynamics of the environment. The agent must explore and discover this information through its interactions, often dealing with uncertainty and incomplete information while trying to achieve its goals.

### **Hill-Climbing Algorithm**

<https://www.scaler.com/topics/hill-climbing-in-ai/>

<https://www.knowledgehut.com/blog/data-science/hill-climbing-algorithm-ai>

<https://www.javatpoint.com/hill-climbing-algorithm-in-ai>

### **Give some real-world examples where the search agents can be used.**

Search Agents are just one kind of algorithm in Artificial Intelligence. Here, an AI has to choose from a large solution space, given that it has a large action space on a large state space. This involves formulating the problem, that the AI is going to solve, in the right way. Some of the real-world examples, where they are being used are:

- 1. Route finding problem:** Examples of applications includes tools for driving directions on websites, in-car systems, etc.
- 2. Travelling salesman problem:** Find the shortest tour to visit each city exactly once.
- 3. VLSI Layout:** position millions of components and connections on a chip to minimize area, and shorten delays.
- 4. Robot Navigation:** Special case of route finding for robots with no specific routes or connections, where state space and action space are potentially infinite.
- 5. Automatic assembly sequencing:** Find an order in which to assemble parts of an object which is a difficult and expensive geometric search.
- 6. Protein design:** Find a sequence of amino acids that will fold into a 3D protein with the right properties to cure some diseases.

### **Can you explain the A\* algorithm and how it uses both heuristic information and cost information to find the optimal path?**

A\* algorithm is an informed search algorithm that uses both heuristic information and cost information to find the optimal path between a start state and a goal state. The algorithm maintains a priority queue of nodes to explore, where the priority is based on the sum of the cost of the path from the start state to the current node and an estimate of the cost from the current node to the goal state.

The estimate of the cost from the current node to the goal state is provided by a heuristic function, which provides an estimate of the distance between the current node and the goal node. The heuristic function should be admissible, meaning that

it should never overestimate the actual cost of reaching the goal node. If the heuristic function is admissible, A\* algorithm is guaranteed to find the optimal path from the start state to the goal state.

At each step, the algorithm selects the node with the lowest priority from the priority queue and expands it, generating all its successor nodes. For each successor node, the algorithm computes the cost of the path from the start state to the successor node and estimates the cost of reaching the goal state from the successor node using the heuristic function. The algorithm then updates the priority of the successor node in the priority queue based on the sum of these two costs.

If the successor node is already in the priority queue, the algorithm checks whether the cost of the path to the successor node is less than the cost previously computed for the node. If so, the algorithm updates the priority of the node in the queue to reflect the lower cost.

The algorithm continues to expand nodes and update their priorities in the queue until it reaches the goal node. When the goal node is reached, the algorithm reconstructs the path from the start state to the goal state by following the parent pointers of the nodes along the path.

In summary, A\* algorithm is an informed search algorithm that combines cost information and heuristic information to find the optimal path between a start state and a goal state. By using an admissible heuristic function, A\* algorithm can guarantee to find the optimal solution efficiently.

### **What are some examples of informed search algorithms that use heuristic information to guide their search?**

There are several examples of informed search algorithms that use heuristic information to guide their search. Here are a few:

**1. A\* Search:** This is perhaps the most well-known informed search algorithm that uses heuristic information to guide its search. search is guaranteed to find the optimal path to the goal if the

heuristic function is admissible.

**2. Greedy Best-First Search:** This algorithm always expands the node that appears to be closest to the goal state based on the heuristic estimate, but it does not consider the cost of the path to the node.

**3. Iterative-Deepening A\*:** This algorithm combines the benefits of A' search and depth-first search by doing a series of depth-first searches with progressively increasing depth limits, where the depth limit at each iteration is determined by the heuristic estimate.

**4. IDA\* Search:** This algorithm is similar to Iterative-Deepening A', but it uses an admissible heuristic function to guide its search and only expands nodes whose estimated cost plus the actual cost to reach the node is less than a certain threshold.

**5. Beam Search:** This algorithm maintains a fixed number of nodes at each level of the search tree based on the heuristic estimate. It only expands nodes that are among the best candidates based on the heuristic estimate, which can lead to faster convergence to a solution.

These are just a few examples of informed search algorithms that use heuristic information to guide their search. There are many other algorithms that use heuristics, and the choice of algorithm depends on the specific problem domain and the quality of available heuristic information.

**What are some advantages and disadvantages of using an on-line search agent, and in what situations might it be the preferred approach?**

**Advantages of using an online search agent include:**

**1. Adaptability:** Online search agents can adapt to changes in the environment and learn from their interactions. They can adjust their behaviour in real-time based on the feedback they receive from the environment.

**2. Real-time decision making:** Online search agents can make decisions quickly and efficiently in real-time. This is particularly important in time-sensitive applications such as robotics, gaming, and autonomous vehicles.

**3. Robustness:** Online search agents are typically designed to be robust to noise, uncertainty, and incomplete information. They can handle unexpected events and recover from errors.

**Disadvantages of using an online search agent include:**

**1. Limited exploration:** Online search agents must balance exploitation (taking actions that are likely to lead to immediate rewards) and exploration (taking actions that may lead to new information).

In some cases, online search agents may become stuck in a suboptimal solution if they do not explore enough.

**2. Lack of complete information:** Online search agents do not have complete information about the environment or the problem. They must make decisions based on incomplete or uncertain information.

**3. Complexity:** Online search agents can be complex to design and implement. They require sophisticated algorithms and data structures to handle the dynamic nature of the environment.

Online search agents are generally preferred in situations where the environment is dynamic and changing, and the agent must make decisions in real-time. Examples include robotics, gaming, and autonomous vehicles. They are also useful in situations where the agent has limited prior knowledge of the environment, and must learn through interaction and exploration. In contrast, offline search agents are preferred in situations where the environment is static and well-understood, and the agent can take the time to explore and plan its actions before execution. Examples include puzzle-solving and pathfinding problems.

### **What is the Constraint satisfaction problem? On what factors does Constraint satisfaction depend?**

Constraint satisfaction is a technique where a problem is solved when its values satisfy certain constraints or rules of the problem. Such type of technique leads to a deeper understanding of the problem structure as well as its complexity.

Constraint satisfaction depends on three components, namely:

X: It is a set of variables.

D: It is a set of domains where the variables reside. There is a specific domain for each variable.

C: It is a set of constraints that are followed by a set of variables.

In constraint satisfaction, domains are the spaces where the variables reside, following the problem-specific constraints. These are the three main elements of a constraint satisfaction technique. The constraint value consists of a pair of {scope, rel}. The scope is a tuple of variables that participate in the constraint and rel is a relation that includes a list of values that the variables can take to satisfy the constraints of the problem.

\*\*

A **Constraint Satisfaction Problem** in artificial intelligence involves a set of variables, each of which has a domain of possible values, and a set of constraints that define the allowable combinations of values for the variables. The goal is to find a value for each variable such that all the constraints are satisfied.

More formally, a CSP is defined as a triple  $(X,D,C)$ , where:

- $X$  is a set of variables  $\{x_1, x_2, \dots, x_n\}$ .
- $D$  is a set of domains  $\{D_1, D_2, \dots, D_n\}$ , where each  $D_i$  is the set of possible values for  $x_i$ .
- $C$  is a set of constraints  $\{C_1, C_2, \dots, C_m\}$ , where each  $C_i$  is a constraint that restricts the values that can be assigned to a subset of the variables.

The goal of a CSP is to find an assignment of values to the variables that satisfies all the constraints. This assignment is called a solution to the CSP.

### Types of Constraints in CSP

Several types of constraints can be used in a Constraint satisfaction problem in artificial intelligence, including:

- **Unary Constraints:**  
A unary constraint is a constraint on a single variable. For example, Variable A not equal to "Red".
- **Binary Constraints:**  
A binary constraint involves two variables and specifies a constraint on their



values. For example, a constraint that two tasks cannot be scheduled at the same time would be a binary constraint.

- **Global Constraints:**

Global constraints involve more than two variables and specify complex relationships between them. For example, a constraint that no two tasks can be scheduled at the same time if they require the same resource would be a global constraint.

### **Give some real world example of constraint satisfaction problem**

1. **Sudoku:** Each cell in the grid must be filled with a number from 1 to 9, ensuring that no number repeats in any row, column, or 3x3 sub grid.
2. **Scheduling:** Assigning timeslots for classes in a school or university, ensuring no conflicts such as overlapping classes for the same student or double-booking a room.
3. **Timetabling:** Creating exam schedules for students, ensuring no student has overlapping exams and spreading exams evenly to avoid student overload.
4. **Map Coloring:** Coloring regions on a map such that no two adjacent regions have the same color, often used in political districting.
5. **Vehicle Routing:** Planning routes for delivery trucks to ensure all deliveries are made within constraints like time windows, vehicle capacities, and minimizing travel distance.
6. **Job Shop Scheduling:** Assigning start times to a set of jobs to be performed on machines, ensuring no two jobs use the same machine simultaneously and meeting job precedence requirements.
7. **Cryptarithms:** Puzzles where letters stand for digits, and you must find the correct digits to satisfy the arithmetic operations (e.g., SEND + MORE = MONEY).

What are the three components of a CSP formulation?

The three components of a CSP formulation are:

**1. Variables:** Variables are the unknowns in the problem that need to be assigned values. Each variable has a domain of possible values it can take on.

**2. Constraints:** Constraints are the conditions that must be satisfied between pairs of variables. They restrict the possible combinations of variable assignments and can be either hard or soft constraints.

**3. Objective function:** The objective function is a measure of how good a solution is. It is used to evaluate the quality of a solution and determine the optimal solution when there are multiple solutions that satisfy all the constraints. The objective function can be either minimization or maximization, depending on the problem.

These three components work together to define the problem space and guide the search for a solution that satisfies all the constraints.

### **Backtracking Search for CSP in Artificial Intelligence:**

Backtracking is a widely used technique for solving CSPs. It is a systematic search algorithm that explores possible assignments for variables, backtracking when it encounters constraints that cannot be satisfied. The algorithm follows these steps:

- Choose an unassigned variable.
- Select a value from its domain.
- Check if the assignment violates any constraints.
- If a constraint is violated, backtrack to the previous variable and try another value.
- Continue this process until all variables are assigned values, or a valid solution is found.

### **Local Search for CSP**

Local search is a type of algorithm used to solve Constraint Satisfaction Problems (CSPs) by iteratively improving a candidate solution until a satisfactory solution is found.

The basic idea behind local search is to start with an initial solution, which may or may not be feasible, and then explore the space of solutions in the neighbourhood

of the current solution. The neighbourhood is defined as the set of solutions that can be reached from the current solution by making a small change. The algorithm then selects the best solution in the neighbourhood and repeats the process until a satisfactory solution is found.

The local search approach has the advantage of being able to handle very large problem spaces, since it only considers a small subset of the solution space at any given time. This can make it more efficient than other methods that require the entire solution space to be searched.

One common type of local search algorithm for CSPs is called Hill Climbing. In this algorithm, the search starts with an initial solution and iteratively explores neighbouring solutions that are better than the current one. If a better solution is found, it becomes the new current solution, and the search continues from there. However, if no better solution is found, the search terminates and the current solution is returned as the best solution found so far.

Another type of local search algorithm is called Simulated Annealing. In this algorithm, the search starts with an initial solution and iteratively explores neighbouring solutions that may be worse than the current one. The algorithm accepts some of these worse solutions based on a probability function that gradually reduces over time, allowing the search to escape from local optima and find a better overall solution.

Local search algorithms can be very effective in solving CSPs, but they do have some limitations. One potential issue is that they can get stuck in local optima, meaning that they find a good solution in the local neighbourhood but fail to explore other parts of the solution space that may contain even better solutions. To address this problem, some local search algorithms incorporate randomization or restart strategies to encourage exploration of the full solution space.

<https://www.almbetter.com/bytes/tutorials/artificial-intelligence/local-search-algorithm-in-artificial-intelligence>

**What is breadth-first search, and how does it differ from depth-first search?**

**What are some situations where BFS would be a better choice than DFS?**

Breadth-first search (BFS) and depth-first search (DFS) are two popular uninformed search algorithms used in artificial intelligence and computer science to traverse and explore all the nodes of a graph or tree data structure.

BFS starts at the root node and explores all the neighbouring nodes at the current depth level before moving on to the next level. It uses a queue data structure to keep track of the nodes that have been visited and their adjacent nodes that have not yet been visited. In BFS, the search progresses level by level, from the shallowest level to the deepest level.

DFS, on the other hand, explores the deepest path first and backtracks if no solution is found. It uses a stack data structure to keep track of the nodes that have been visited and their adjacent nodes that have not yet been visited. In DFS, the search progresses deeper and deeper into the graph or tree, following each path as far as possible before backtracking.

**Some situations where BFS would be a better choice than DFS include:**

- 1) Finding the shortest path:** BFS guarantees that it will find the shortest path between the starting node and the goal node if one exists. This makes it a better choice than DFS when the shortest path is important.
- 2) Finding the optimal solution:** If the cost of each edge is the same, BFS is optimal, meaning that it will find the solution with the minimum number of steps.
- 3) When the search space is not very large:** BFS requires more memory to store the queue of nodes to be explored at the next level, so it may not be the best choice when the search space is very large.
- 4) When the solution is close to the starting node:** If the solution is close to the starting node, BFS will find it quickly, whereas DFS may take longer as it searches deeper into the graph or tree.

In summary, BFS and DFS are two common search algorithms with different advantages and disadvantages. BFS is better for finding the shortest path or optimal solution, while DFS is better for exploring deep paths in a graph or tree. The choice of algorithm depends on the specific problem being solved and the characteristics of the search space.

## Alpha-Beta Pruning

<https://www.scaler.com/topics/artificial-intelligence-tutorial/alpha-beta-pruning/>

<https://www.almabetter.com/bytes/tutorials/artificial-intelligence/alpha-beta-pruning>

<https://www.javatpoint.com/ai-alpha-beta-pruning>

### What are the advantages and disadvantages of alpha-beta pruning?

Using Alpha-Beta pruning is always beneficial, as it offers various benefits like better time complexity, over the minimax algorithm. However, there are still some drawbacks associated with this algorithm, which prevent it from being the ideal or the go-to search algorithm for all.

#### Advantages:

1. Allows elimination of the search tree branches.
2. Limits the search time to more promising sub-trees, which enables a deeper search.
3. Reduces computation and searching during the minimax algorithm.
4. Prevents the use of additional computational time, making the process more responsive and faster.

#### Disadvantages:

1. It does not solve all the problems associated with the original minimax algorithm.
2. Requires a set depth limit, as in most cases, it is not feasible to search the entire game tree.
3. Though designed to calculate the good move, it also calculates the values of all the legal moves.

## **Adversarial Search**

Adversarial search in artificial intelligence is used to resolve two-person games in which one player seeks to maximize their score while the other tries to minimize it. Chess, checkers, and tic-tac-toe are the three most popular games that can be solved via adversarial search. In many real-world applications, such as financial decision-making and military operations, adversarial search is used. To give players clever opponents in video games, it is also deployed.

### **Role of Adversarial Search in AI:**

The role of adversarial search in AI is to model and navigate scenarios where decision-making involves competing entities with opposing goals. It plays a crucial role in several aspects:

- 1. Game-Playing:** Adversarial search is prominently used in AI for playing games. Whether it's classic board games like chess and checkers or modern video games, AI agents employ adversarial search techniques to make strategic decisions and outmaneuver human or computer opponents.
- 2. Decision-Making:** Beyond games, adversarial search has real-world applications in decision-making processes. For example, in economics, it is used to model competitive markets and strategic interactions between companies. In robotics, it assists autonomous agents in planning their actions while considering the intentions and movements of potential adversaries.

### **What are the different game scenarios using adversarial search?**

The different game scenarios are:

#### **1. Games with Perfect information**

These are open and transparent games where each player has all complete information on the status of the game, and the move of the opponent is quite visible to the players. Chess, Checkers, Ethello and Go are some of such games.

#### **2. Games with Imperfect information**

In these games, players will not have any information on the status of the game, and it will have blind moves by the players assuming certain conditions and predicting the outcomes. Tic-Tac-Toe, Battleship, Blind and Bridge are some examples of this type of game.

### **3. Deterministic Games**

These games follow pre-defined rules and pattern, and it is played in a transparent manner. Each player can see the moves of the opponents clearly. Chess, Checkers, Tic-tac-toe, Go are some of the games played in this manner.

### **4. Non-Deterministic Games**

Luck, Chance, Probability plays a major role in these types of game. The outcome of these games is highly unpredictable, and players will have to take random shots relying on luck. Poker, Monopoly, Backgammon are few of the games played in this method.

### **5. Zero-Sum games**

These games are purely competitive in nature, and one player's gain is compensated by other player's loss. The net value of gain and loss is zero, and each player will have different conflicting strategies in these games. Depending on the game environment and game situation, each player will either try to maximize the gain or minimize the loss.

*Adversial Search, Min-Max, Alpha-Beta*

<https://www.almabetter.com/bytes/tutorials/artificial-intelligence/adversarial-search-in-artificial-intelligence>

### **Knowledge-Based Agent**

Knowledge-based agents are artificial intelligence systems that use knowledge to perform their tasks. They are composed of two main parts, a knowledge base, and an inference system, and they operate by making deductions, decisions, and conclusions based on the available knowledge.

### **Knowledge-Based Agents are Composed of Two Main Parts**

#### **Knowledge-Base**

The knowledge base of a knowledge-based agent is a collection of knowledge that the agent uses to make decisions. The knowledge can be explicit, such as rules or facts, or implicit, such as relationships or patterns. The knowledge base is stored in a database or a knowledge representation system.

## Inference System

The inference system of a knowledge-based agent is responsible for using the knowledge in the knowledge base to make decisions. The inference system uses reasoning methods, such as deduction, induction, and abduction, to infer new knowledge from existing knowledge.

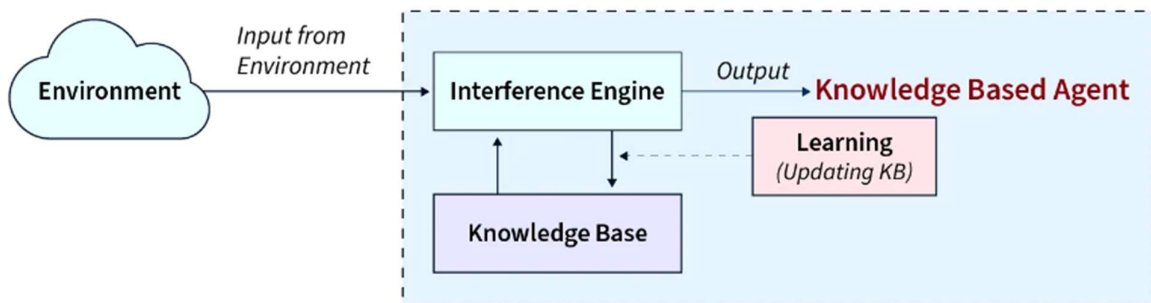
## The Architecture of Knowledge-Based Agent

The diagram below depicts a general architecture for a knowledge-based agent (KBA):

The KBA receives input from the environment through perception, which the inference engine processes.

The inference engine communicates with the knowledge base (KB) to determine the appropriate action based on the knowledge stored in the KB.

The learning element of the KBA regularly updates the KB by incorporating new knowledge.



The KB is a central component of the KBA, consisting of a collection of sentences expressed in a knowledge representation language. These sentences represent facts about the world and are stored in the KB for the KBA to access and utilize.

## Why Use a Knowledge Base?

A knowledge base is used in a knowledge-based agent to provide a structured way to store and retrieve information. The knowledge base allows the agent to make decisions based on available knowledge rather than relying on hard-coded rules. Additionally, the knowledge base can be updated as new information becomes available, making the agent more adaptable to changing situations.

## Operations Performed by KBA



There are three main operations that a knowledge-based agent (KBA) performs to demonstrate intelligent behaviour. The first operation is called **TELL**, where the KBA informs the knowledge base about the information it has perceived from the environment.

The second operation is called **ASK**, where the KBA requests the knowledge base to suggest appropriate action.

The third operation is **PERFORM**, where the KBA executes the selected action.

**Describe the Wumpus World as an example world used in artificial intelligence. What are the features of the environment, and what are the goals of the agent operating in this world?**

The Wumpus World is a well-known example world used in artificial intelligence to test various intelligent agent algorithms. It is a simple, grid-based environment in which an agent must navigate to find gold while avoiding dangerous pits and a hostile creature called the Wumpus. The environment is represented as a two-dimensional grid, with each cell of the grid being either empty, containing a pit, containing gold, or containing the Wumpus.

The Wumpus is stationary and does not move, but if the agent enters a cell with the Wumpus, the agent is eaten and the game ends. The pits are also stationary and cannot be moved, and if the agent falls into a pit, the game ends. The gold is located in a single cell of the environment, and the goal of the agent is to find the gold and return to the starting position.

The agent has a limited sensor range and can only perceive the cells immediately adjacent to its current position. This means that the agent must explore the environment and use its knowledge of the cells it has already visited to make informed decisions about where to move next. The agent also has a limited supply of arrows, which can be used to kill the Wumpus from a distance.
















The goal of the agent in the Wumpus World is to find the gold and return to the starting position while avoiding the pits and the Wumpus. The agent must use its knowledge of the environment, such as the locations of the pits and the Wumpus, to plan its actions and avoid danger. The Wumpus World is a popular example world used in artificial intelligence because it is simple enough to be understood easily, but complex enough to challenge intelligent agent algorithms.

**Wumpus World Cave Problem**

The Wumpus world in AI is a cave with four chambers linked by passageways. So there are a total of 16 chambers that are linked to one another. We now have a knowledge-based agent who will advance in this universe. The cave has a chamber with a beast named Wumpus, who eats anyone who enters it. The agent can shoot the wumpus, but the agent only has one projectile. Some pit rooms in the Wumpus world in AI are bottomless, and if the agent falls into one of them, he will be stuck there eternally. The exciting aspect of this cave is discovering a heap of gold in one of its rooms. So the agent's objective is to locate the gold and climb out of the cave without being eaten by wumpus or falling into Pits. The agent will be rewarded if he returns with gold, but he will be punished if he is eaten by wumpus or slips into the pit.

Some elements can assist the agent in navigating the tunnel. These elements are listed below:

- The rooms adjacent to the Wumpus chamber are stinky, so there will be a stench.
- The room closest to the PITs has a breeze, so if the agent gets close to the PIT, he will notice the breeze.
- Glitter will be present in the chamber if the room contains gold.
- If the agent confronts the wumpus, it can be killed, and the wumpus will scream horribly, which can be heard throughout the cave.

4	 stench		 Breeze	
3	 Wumpus	 Breeze  stench  GOLD		 Breeze
2	 stench		 Breeze	
1	 Agent	 Breeze		 Breeze
	1	2	3	4

## PEAS Description of Wumpus World

To build an intelligent agent for the Wumpus World, we must first define the problem's Performance, Environment, Actuators, and Sensors (PEAS).

### 1. Performance:

- +1000 bonus points if the agent returns from the tunnel with the gold.
- Being eaten by the wumpus or plummeting into the pit results in a -1000 point penalty.
- Each move is worth -1, and using an arrow is worth -10.
- The game is over if either agent dies or exits the tunnel.

### 2. Environment:

- A four-by-four grid of chambers.
- The operative begins in room square [1, 1], facing the right.
- Wumpus and gold locations are selected randomly except for the first square [1,1].
- Except for the first square, each square in the tunnel has a 0.2 chance of being a pit.

3. **Actuators:** They are the actions that the agent can take to interact with the world. The worker in Wumpus World in AI can carry out the following tasks:
- Left turn
  - Right turn
  - Move forward
  - Grab
  - Release
  - Shoot
4. **Sensors:** They are how the agent senses its surroundings. The agent's instruments in the Wumpus World provide the following information:
- If the agent is in the chamber next to the wumpus, he will notice the stench. (Not diagonally).
  - If the agent is in the room immediately adjacent to the pit, he will notice a breeze.
  - The agent will notice the glitter in the chamber with the gold.
  - The agent will notice the bump if he runs into a wall.
  - When the Wumpus is shot, it lets out a horrifying scream that can be heard throughout the tunnel.
  - These perceptions can be represented as a five-element list with distinct indicators for each sensor.
  - For example, if an agent detects stench and breeze but not glitter, bump, or scream, it can be depicted as [Stench, Breeze, None, None].

<https://www.scaler.com/topics/artificial-intelligence-tutorial/wumpus-world-in-ai/>

## Propositional Logic

Propositional logic is a fundamental part of artificial intelligence and computer science. It is a branch of mathematical logic that deals with the study of propositions, their truth values, and the logical relationships that exist between them. Propositional logic (PL) is the simplest form of logic where all the statements

are made by propositions. A proposition is a declarative statement which is either true or false.

### **Following are some basic facts about propositional logic:**

- Propositional logic is a formal language that uses symbols to represent propositions and logical connectives to combine them. The symbols used in propositional logic include letters such as  $p$ ,  $q$ , and  $r$ , which represent propositions, and logical connectives such as  $\wedge$  (conjunction),  $\vee$  (disjunction), and  $\neg$  (negation), which are used to combine propositions.
- Propositions can be either true or false, but it cannot be both.
- Logical connectives are used to combine propositions to form more complex statements.

### **Syntax of Propositional Logic**

Syntax of propositional logic refers to the formal rules for constructing statements in propositional logic. Propositional logic deals with the study of propositions, which are declarative statements that are either true or false. The syntax of propositional logic consists of two main components: atomic propositions and compound propositions.

#### **Atomic Propositions**

Atomic propositions are simple statements that cannot be broken down into simpler statements. They are the building blocks of propositional logic. An atomic proposition can be represented by a letter or symbol, such as  $p$ ,  $q$ ,  $r$ , or  $s$ . For example, the following are atomic propositions:

$p$ : The sky is blue.  $q$ : The grass is green.  $r$ :  $2+2=4$ .  $s$ : The Earth orbits the Sun.

#### **Compound Propositions**

Compound propositions are formed by combining atomic propositions using logical operators. There are several logical operators in propositional logic, including negation, conjunction, disjunction, implication, and equivalence.

Example:

- "It is raining today, and state it is wet."

### **Logical Connectives**

When connecting two simpler assertions or logically expressing a statement, logical connectives are used. Using logical connectives, we can build compound propositions. The following list of connectives includes the main five:

### **Negation**

The negation of a proposition  $p$  is denoted by  $\neg p$  and is read as "not  $p$ ". For example:  $\neg p$ : The sky is not blue.

### **Conjunction**

The conjunction of two propositions  $p$  and  $q$  is denoted by  $p \wedge q$  and is read as " $p$  and  $q$ 's". The conjunction is true only if both  $p$  and  $q$  are true. For example:  $p \wedge q$ : The sky is blue and the grass is green.

### **Disjunction**

The disjunction of two propositions  $p$  and  $q$  is denoted by  $p \vee q$  and is read as " $p$  or  $q$ ". The disjunction is true if at least one of  $p$  and  $q$  is true. For example:  $p \vee q$ : The sky is blue or the grass is green.

### **Implication**

The implication of two propositions  $p$  and  $q$  is denoted by  $p \rightarrow q$  and is read as "if  $p$  then  $q$ ". The implication is false only if  $p$  is true

### **Biconditional**

A sentence such as  $P \Leftrightarrow Q$  is a Biconditional sentence, example I will eat lunch if and only if my mood improves.  $P$ = I will eat lunch,  $Q$ = if my mood improves, it can be represented as  $P \Leftrightarrow Q$ .

## **Truth Tables for Different Logical Operators (AND, OR, NOT):**

Here are truth tables for the basic logical operators in propositional logic:

### **1. Conjunction (AND, $\wedge$ ) Truth Table:**

<b>P</b>	<b>Q</b>	<b><math>P \wedge Q</math></b>
True	True	True
True	False	False
False	True	False

P	Q	$P \wedge Q$
False	False	False

## 2. Disjunction (OR, $\vee$ ) Truth Table:

P	Q	$P \vee Q$
True	True	True
True	False	True
False	True	True
False	False	False

## 3. Negation (NOT, $\neg$ ) Truth Table:

P	$\neg P$
True	False
False	True

These truth tables illustrate how the logical operators evaluate the truth value of compound propositions based on the truth values of their constituent propositions. They provide a formal and systematic way to determine the truth value of more complex statements in propositional logic.

## How Propositional Logic is Used for Knowledge Representation in AI

Propositional logic plays a crucial role in knowledge representation within the field of artificial intelligence (AI). Knowledge representation is the process of structuring information in a way that AI systems can understand, manipulate, and reason about it. Propositional logic is one of the fundamental techniques used for this purpose. Here's how it's used for knowledge representation in AI:

**1. Encoding Facts:** Propositional logic is employed to represent facts or assertions about the world. These facts are typically expressed as propositions, which can be true or false. For example, in a medical diagnosis system, "Patient has a fever" might be represented as a proposition  $P$ , which is true or false based on the patient's actual condition.

**2. Modeling Relationships:** Propositional logic allows the representation of relationships between facts. Logical operators (AND, OR) can be used to express complex relationships. For instance, "Patient has a fever AND cough" is represented as  $P \wedge C$ , where  $P$  and  $C$  are propositions.

**3. Inference and Reasoning:** AI systems can use propositional logic to make inferences or draw conclusions. For example, if the knowledge base contains the proposition "Patient has a fever AND cough" ( $P \wedge C$ ), and the system knows that "Fever implies flu" ( $P \rightarrow F$ ) and "Cough implies flu" ( $C \rightarrow F$ ), it can infer that the patient likely has the flu ( $F$ ).

**4. Decision-Making:** Knowledge represented in propositional logic allows AI systems to make decisions based on a set of rules. By evaluating the truth values of propositions and applying rules, AI systems can choose the best course of action.

<https://www.scaler.com/topics/artificial-intelligence-tutorial/propositional-logic-in-ai/>

<https://www.almabetter.com/bytes/tutorials/artificial-intelligence/propositional-logic-in-ai>

**Describe the process of reasoning in propositional logic, including the various reasoning patterns that can be used to draw conclusions from a set of propositions. Provide examples of how these reasoning patterns can be applied in the context of a specific problem or domain.**



Propositional logic is a branch of logic that deals with propositions, which are statements that can be either true or false. Reasoning in propositional logic involves drawing conclusions based on a set of propositions, using various reasoning patterns. The basic reasoning patterns in propositional logic include modus ponens, modus tollens, hypothetical syllogism, disjunctive syllogism, and addition.

Modus ponens is a pattern of reasoning that involves inferring a consequent from a conditional statement and its antecedent. For example, if we have the conditional statement "If it is raining, then the streets are wet" and the proposition "It is raining," we can use modus ponens to infer the consequent "The streets are wet." This pattern of reasoning is useful in many domains, such as weather forecasting, where we can use conditional statements to predict the outcome of certain events.

Modus tollens is a pattern of reasoning that involves inferring the negation of the antecedent from the negation of a conditional statement and its consequent. For example, if we have the conditional statement "If it is raining, then the streets are wet" and the proposition "The streets are not wet," we can use modus tollens to infer the negation of the antecedent "It is not raining." This pattern of reasoning is useful in many domains, such as law enforcement, where we can use conditional statements to infer the likelihood of certain events.

Hypothetical syllogism is a pattern of reasoning that involves inferring a conclusion from two conditional statements, where the consequent of the first statement is the antecedent of the second statement. For example, if we have the conditional statements "If it is raining, then the streets are wet" and "If the streets are wet, then people are using umbrellas," we can use hypothetical syllogism to infer the conclusion "If it is raining, then people are using umbrellas." This pattern of reasoning is useful in many domains, such as transportation, where we can use conditional statements to infer the effects of certain events.

Disjunctive syllogism is a pattern of reasoning that involves inferring the negation of one disjunct from the affirmation of the other disjunct in a disjunctive proposition. For example, if we have the disjunctive proposition "Either it is raining or the sun is shining" and the proposition "It is not raining," we can use disjunctive syllogism to infer "The sun is shining." This pattern of reasoning is useful in many domains, such as travel, where we can use disjunctive propositions to infer the availability of different modes of transportation.

Addition is a pattern of reasoning that involves inferring a proposition from the conjunction of two propositions. For example, if we have the propositions "It is

raining" and "The streets are wet," we can use addition to infer the proposition "It is raining and the streets are wet." This pattern of reasoning is useful in many domains, such as agriculture, where we can use conjunctions to infer the presence of certain conditions that are necessary for crop growth.

In summary, reasoning in propositional logic involves drawing conclusions based on a set of propositions using various reasoning patterns, such as modus ponens, modus tollens, hypothetical syllogism, disjunctive syllogism, and addition. These reasoning patterns are useful in many domains, such as weather forecasting, law enforcement, transportation, travel, and agriculture.

## First-Order Logic

**First Order Logic** in Artificial Intelligence is a technique used for **knowledge representation**. It is an extension of propositional logic and unlike propositional logic, it is sufficiently expressive in representing any natural language construct. First Order Logic in AI is also known as **Predicate Logic** or **First Order Predicate Logic**. It is a robust technique to represent objects as well as their relationships. Unlike propositional logic, First Order Logic in Artificial Intelligence doesn't only include facts but also different other entities as listed below.

1. **Objects:**

Objects can denote any real-world entity or any variable.  
E.g., A, B, colours, theories, circles etc.

2. **Relations:**

Relations represent the links between different objects. Relations can be unary (relations defined for a single term) and n-ary (relations defined for n terms). E.g., blue, round (unary); friends, siblings (binary); etc.

3. **Functions:**

Functions map their input object to the output object using their underlying relation. Eg: father\_of(), mother\_of() etc.

## Parts of First Order Logic

First-order logic in Artificial Intelligence comprises two main components, which are as follows.

### Syntax:

Syntax represents the rules to write expressions in First Order Logic in Artificial Intelligence.

### Semantics:

Semantics refers to the techniques that we use to evaluate an expression of First Order Logic in AI. These techniques use various known relations and facts of the respective environment to deduce the Boolean value of the given First Order Logic expression.

### Syntax

The **syntax** of FOL decides which collection of symbols is a logical expression.

The basic syntactic elements of FOL are symbols. We use symbols to write statements in shorthand notation.

### Basic elements of FOL

Name	Symbol
Constant	1, 6, A,W,New York, Elie, Dog...
Variables	a, b, c, x, y, z...
Predicates	<, >, brother, sister, father...
Equality	==
Function	Sqrt, LessThan, Sin( $\theta$ )...
Quantifier	$\forall$ , $\exists$
Connectives	$\wedge$ , $\vee$ , $\neg$ , $\Rightarrow$ , $\Leftrightarrow$

### Atomic and complex sentences in FOL

#### 1. Atomic Sentence

- This is a basic sentence of FOL formed from a predicate symbol followed by a parenthesis with a sequence of terms.
- We can represent atomic sentences as a predicate (value1, value2...., value n).

## Example

1. John and Michael are colleagues → Colleagues (John, Michael)
2. German Shepherd is a dog → Dog (German Shepherd)

## 2. Complex sentence

**Complex sentences** are made by combining atomic sentences using connectives.

FOL is further divided into two parts:

- **Subject:** the main part of the statement.
- **Predicate:** defined as a relation that binds two atoms together.

## Example

1. Colleague (Oliver, Benjamin)  $\wedge$  Colleague (Benjamin, Oliver)
2. "x is an integer"

It has two parts;

- first, x is the subject.
- second, "is an integer" is called a predicate.

<https://www.scaler.com/topics/artificial-intelligence-tutorial/first-order-logic-in-ai/>

<https://www.educative.io/answers/what-is-first-order-logic-in-artificial-intelligence>

## What is the use of first order logic?

First-order logic (FOL) is a powerful formal system used in various domains to represent and reason about propositions involving objects and their relationships. Here are some of its key uses:

1. **Knowledge Representation:** FOL is used to model and represent complex structures and relationships in a precise and unambiguous way, which is essential for fields like artificial intelligence and semantic web technologies.
2. **Automated Reasoning:** It provides a foundation for building algorithms that can automatically infer new information from given facts and rules, which is crucial for developing intelligent systems and proving mathematical theorems.
3. **Database Query Languages:** The principles of FOL underpin the design of query languages like SQL, enabling complex querying and manipulation of databases.

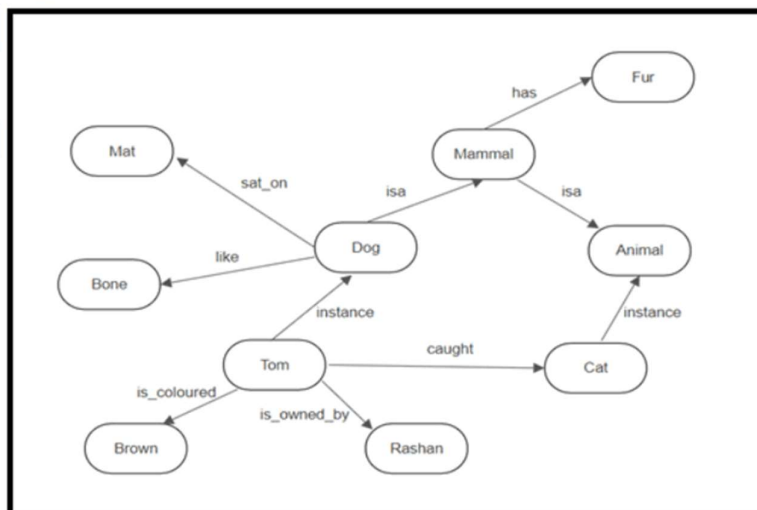
4. **Natural Language Processing:** FOL helps in understanding and generating human language by representing the meaning of sentences and their logical relationships, aiding in tasks such as machine translation, information extraction, and dialogue systems.
5. **Formal Verification:** FOL is used to verify the correctness of hardware and software systems by formally proving that they meet their specifications, ensuring reliability and safety in critical applications.
6. **Expert Systems:** FOL provides a basis for encoding expert knowledge and rules in systems that perform tasks such as medical diagnosis, legal reasoning, and financial decision-making.

These applications demonstrate how FOL is essential for capturing, manipulating, and reasoning about knowledge in a structured and logical manner across various disciplines.

### What is semantic net?

In the domain of Artificial Intelligence, a *semantic network* serves as a graphical representation of knowledge in the form of interconnected nodes and links. These connections depict the relationships between different concepts, providing a structured framework for organizing and processing information. With a primary focus on enhancing data understanding and knowledge representation, semantic networks have become instrumental in various AI applications.

In a nutshell, semantic networks serve as the blueprint of knowledge representation in the world of AI. They offer us a way to organize and navigate the vast seas of data and concepts, providing context and structure to the digital landscape.



<https://www.almabetter.com/bytes/tutorials/artificial-intelligence/semantic-network-in-ai>

[https://www.larksuite.com/en\\_us/topics/ai-glossary/semantic-network](https://www.larksuite.com/en_us/topics/ai-glossary/semantic-network)

**With the help of semantic net, prove that Sourav is 6 feet tall and he is taller than Sachin.**

