What is AI Technique?

In the real world, the knowledge has some unwelcomed properties –

- Its volume is huge, next to unimaginable.
- It is not well-organized or well-formatted.
- It keeps changing constantly.

AI Technique is a manner to organize and use the knowledge efficiently in such a way that –

- It should be perceivable by the people who provide it.
- It should be easily modifiable to correct errors.
- It should be useful in many situations though it is incomplete or inaccurate.

AI techniques elevate the speed of execution of the complex program it is equipped with.

Applications of AI

AI has been dominant in various fields such as –

- **Gaming** – AI plays crucial role in strategic games such as chess, poker, tic-tac-toe, etc., where machine can think of large number of possible positions based on heuristic knowledge.
- **Natural Language Processing** – It is possible to interact with the computer that understands natural language spoken by humans.
- **Expert Systems** – There are some applications which integrate machine, software, and special information to impart reasoning and advising. They provide explanation and advice to the users.
- **Vision Systems** – These systems understand, interpret, and comprehend visual input on the computer. For example,
    - A spying airplane takes photographs, which are used to figure out spatial information or map of the areas.
    - Doctors use clinical expert system to diagnose the patient.
    - Police use computer software that can recognize the face of criminal with the stored portrait made by forensic artist.

- **Speech Recognition** – Some intelligent systems are capable of hearing and comprehending the language in terms of sentences and their meanings while a human talks to it. It can handle different accents, slang words, noise in the background, change in human's noise due to cold, etc.
- **Handwriting Recognition** – The handwriting recognition software reads the text written on paper by a pen or on screen by a stylus. It can recognize the shapes of the letters and convert it into editable text.
- **Intelligent Robots** – Robots are able to perform the tasks given by a human. They have sensors to detect physical data from the real world such as light, heat, temperature, movement, sound, bump, and pressure. They have efficient processors, multiple sensors and huge memory, to exhibit intelligence. In addition, they are capable of learning from their mistakes and they can adapt to the new environment.

What is Artificial Intelligence (AI)?

In today's world, technology is growing very fast, and we are getting in touch with different new technologies day by day.

Here, one of the booming technologies of computer science is Artificial Intelligence which is ready to create a new revolution in the world by making intelligent machines. The Artificial Intelligence is now all around us. It is currently working with a variety of subfields, ranging from general to specific, such as self-driving cars, playing chess, proving theorems, playing music, Painting, etc.

AI is one of the fascinating and universal fields of Computer science which has a great scope in future. AI holds a tendency to cause a machine to work as a human.

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.*"

So, we can define AI as:

> **"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."**

Artificial Intelligence exists when a machine can **have human based skills such as learning, reasoning, and solving problems**

With Artificial Intelligence you do not need to preprogram a machine to do some work, despite that you can create a machine with programmed algorithms which can work with own intelligence, and that is the awesomeness of AI.

It is believed that AI is not a new technology, and some people says that as per Greek myth, there were Mechanical men in early days which can work and behave like humans.

Why Artificial Intelligence?

Before Learning about Artificial Intelligence, we should know that what is the importance of AI and why should we learn it. Following are some main reasons to learn about AI:

- With the help of AI, you can create such software or devices which can solve real-world problems very easily and with accuracy such as health issues, marketing, traffic issues, etc.
- With the help of AI, you can create your personal virtual Assistant, such as Cortana, Google Assistant, Siri, etc.
- With the help of AI, you can build such Robots which can work in an environment where survival of humans can be at risk.
- AI opens a path for other new technologies, new devices, and new Opportunities.

Goals of Artificial Intelligence

Following are the main goals of Artificial Intelligence:

1. Replicate human intelligence
2. Solve Knowledge-intensive tasks
3. An intelligent connection of perception and action
4. Building a machine which can perform tasks that requires human intelligence such as:
    - Proving a theorem
    - Playing chess
    - Plan some surgical operation

- Driving a car in traffic

5. Creating some system which can exhibit intelligent behavior, learn new things by itself, demonstrate, explain, and can advise to its user.

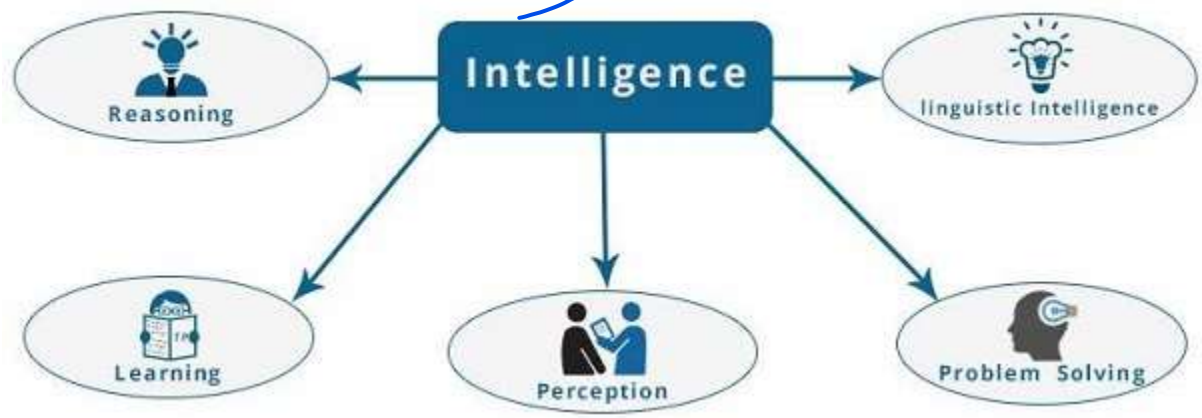
What Comprises to Artificial Intelligence?

Artificial Intelligence is not just a part of computer science even it's so vast and requires lots of other factors which can contribute to it. To create the AI first we should know that how intelligence is composed, so the Intelligence is an intangible part of our brain which is a combination of **Reasoning, learning, problem-solving perception, language understanding, etc**.

What is Intelligence Composed of?

The intelligence is intangible. It is composed of –

- Reasoning
- Learning
- Problem Solving
- Perception
- Linguistic Intelligence



Let us go through all the components briefly –

- **Reasoning** – It is the set of processes that enables us to provide basis for judgement, making decisions, and prediction.

- **Learning** – It is the activity of gaining knowledge or skill by studying, practising, being taught, or experiencing something. Learning enhances the awareness of the subjects of the study.

- **Problem Solving** – It is the process in which one perceives and tries to arrive at a desired solution from a present situation by taking some path, which is blocked by known or unknown hurdles.
  Problem solving also includes **decision making**, which is the process of selecting the best suitable alternative out of multiple alternatives to reach the desired goal are available.

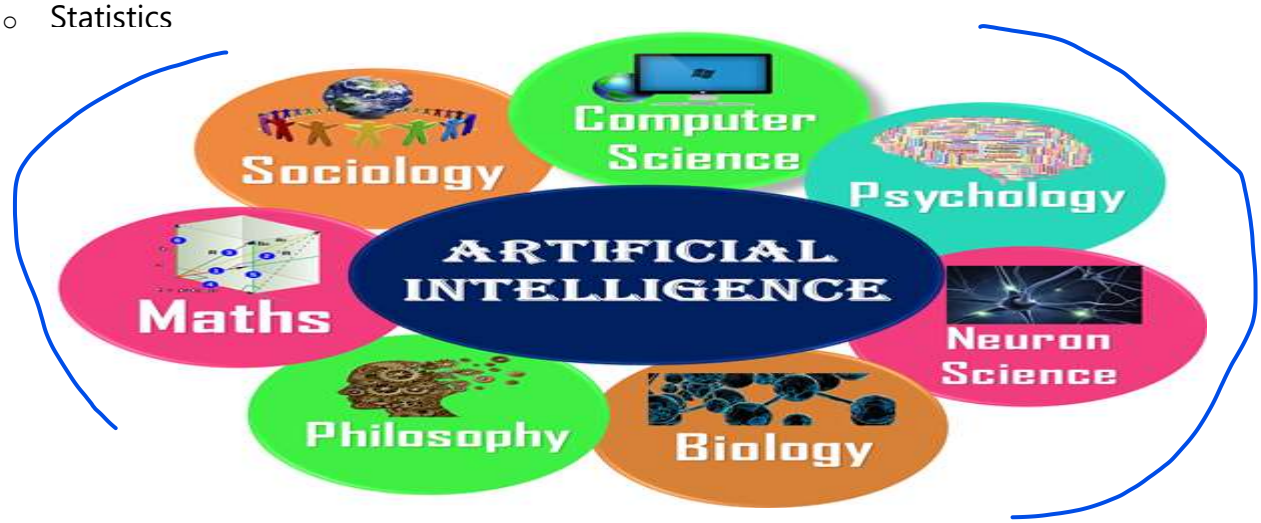
- **Perception** – It is the process of acquiring, interpreting, selecting, and organizing sensory information.
  Perception presumes **sensing**. In humans, perception is aided by sensory organs. In the domain of AI, perception mechanism puts the data acquired by the sensors together in a meaningful manner.

- **Linguistic Intelligence** – It is one's ability to use, comprehend, speak, and write the verbal and written language. It is important in interpersonal communication.

  To achieve the above factors for a machine or software Artificial Intelligence requires the following discipline:

  - Mathematics
  - Biology
  - Psychology
  - Sociology
  - Computer Science
  - Neurons Study

- Statistics



Advantages of Artificial Intelligence

Following are some main advantages of Artificial Intelligence:

- **High Accuracy with less errors:** AI machines or systems are prone to less errors and high accuracy as it takes decisions as per pre-experience or information.
- **High-Speed:** AI systems can be of very high-speed and fast-decision making, because of that AI systems can beat a chess champion in the Chess game.
- **High reliability:** AI machines are highly reliable and can perform the same action multiple times with high accuracy.
- **Useful for risky areas:** AI machines can be helpful in situations such as defusing a bomb, exploring the ocean floor, where to employ a human can be risky.
- **Digital Assistant:** AI can be very useful to provide digital assistant to the users such as AI technology is currently used by various E-commerce websites to show the products as per customer requirement.
- **Useful as a public utility:** AI can be very useful for public utilities such as a self-driving car which can make our journey safer and hassle-free, facial recognition for security purpose, Natural language processing to communicate with the human in human-language, etc.

Disadvantages of Artificial Intelligence

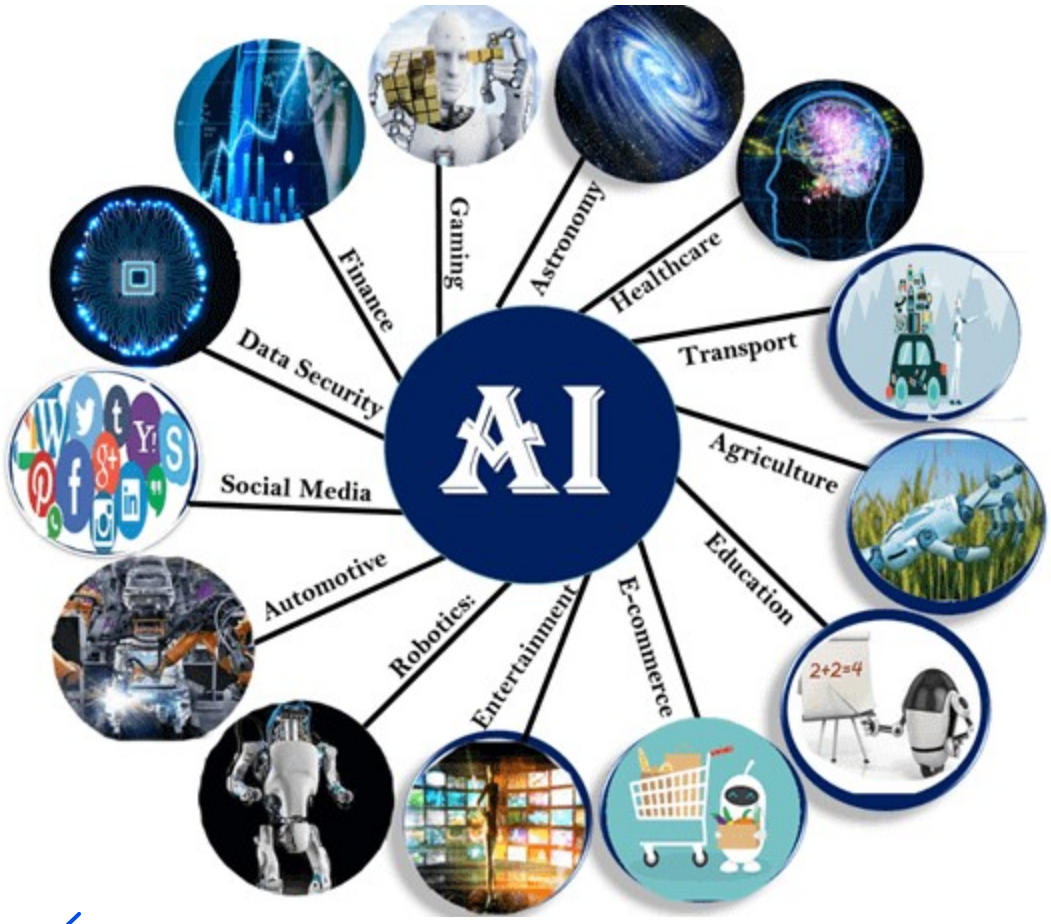
Every technology has some disadvantages, and thesame goes for Artificial intelligence. Being so advantageous technology still, it has some disadvantages which we need to keep in our mind while creating an AI system. Following are the disadvantages of AI:

- **High Cost:** The hardware and software requirement of AI is very costly as it requires lots of maintenance to meet current world requirements.
- **Can't think out of the box:** Even we are making smarter machines with AI, but still they cannot work out of the box, as the robot will only do that work for which they are trained, or programmed.
- **No feelings and emotions:** AI machines can be an outstanding performer, but still it does not have the feeling so it cannot make any kind of emotional attachment with human, and may sometime be harmful for users if the proper care is not taken.
- **Increase dependency on machines:** With the increment of technology, people are getting more dependent on devices and hence they are losing their mental capabilities.
- **No Original Creativity:** As humans are so creative and can imagine some new ideas but still AI machines cannot beat this power of human intelligence and cannot be creative and imaginative.

Application of AI

Artificial Intelligence has various applications in today's society. It is becoming essential for today's time because it can solve complex problems with an efficient way in multiple industries, such as Healthcare, entertainment, finance, education, etc. AI is making our daily life more comfortable and fast.

Following are some sectors which have the application of Artificial Intelligence:



1. AI in Astronomy

- Artificial Intelligence can be very useful to solve complex universe problems. AI technology can be helpful for understanding the universe such as how it works, origin, etc.

2. AI in Healthcare

- In the last, five to ten years, AI becoming more advantageous for the healthcare industry and going to have a significant impact on this industry.
- Healthcare Industries are applying AI to make a better and faster diagnosis than humans. AI can help doctors with diagnoses and can inform when patients are worsening so that medical help can reach to the patient before hospitalization.

3. AI in Gaming

- AI can be used for gaming purpose. The AI machines can play strategic games like chess, where the machine needs to think of a large number of possible places.

4. AI in Finance

- AI and finance industries are the best matches for each other. The finance industry is implementing automation, chatbot, adaptive intelligence, algorithm trading, and machine learning into financial processes.

5. AI in Data Security

- The security of data is crucial for every company and cyber-attacks are growing very rapidly in the digital world. AI can be used to make your data more safe and secure. Some examples such as AEG bot, AI2 Platform, are used to determine software bug and cyber-attacks in a better way.

6. AI in Social Media

- Social Media sites such as Facebook, Twitter, and Snapchat contain billions of user profiles, which need to be stored and managed in a very efficient way. AI can organize and manage massive amounts of data. AI can analyze lots of data to identify the latest trends, hashtag, and requirement of different users.

7. AI in Travel & Transport

- AI is becoming highly demanding for travel industries. AI is capable of doing various travel related works such as from making travel arrangement to suggesting the hotels, flights, and best routes to the customers. Travel industries are using AI-powered chatbots which can make human-like interaction with customers for better and fast response.

8. AI in Automotive Industry

- Some Automotive industries are using AI to provide virtual assistant to their user for better performance. Such as Tesla has introduced TeslaBot, an intelligent virtual assistant.
- Various Industries are currently working for developing self-driven cars which can make your journey more safe and secure.

9. AI in Robotics:

- Artificial Intelligence has a remarkable role in Robotics. Usually, general robots are programmed such that they can perform some repetitive task, but with the help of AI, we can create intelligent robots which can perform tasks with their own experiences without pre-programmed.

- Humanoid Robots are best examples for AI in robotics, recently the intelligent Humanoid robot named as Erica and Sophia has been developed which can talk and behave like humans.

10. AI in Entertainment

- We are currently using some AI based applications in our daily life with some entertainment services such as Netflix or Amazon. With the help of ML/AI algorithms, these services show the recommendations for programs or shows.

11. AI in Agriculture

- Agriculture is an area which requires various resources, labor, money, and time for best result. Now a day's agriculture is becoming digital, and AI is emerging in this field. Agriculture is applying AI as agriculture robotics, solid and crop monitoring, predictive analysis. AI in agriculture can be very helpful for farmers.

12. AI in E-commerce

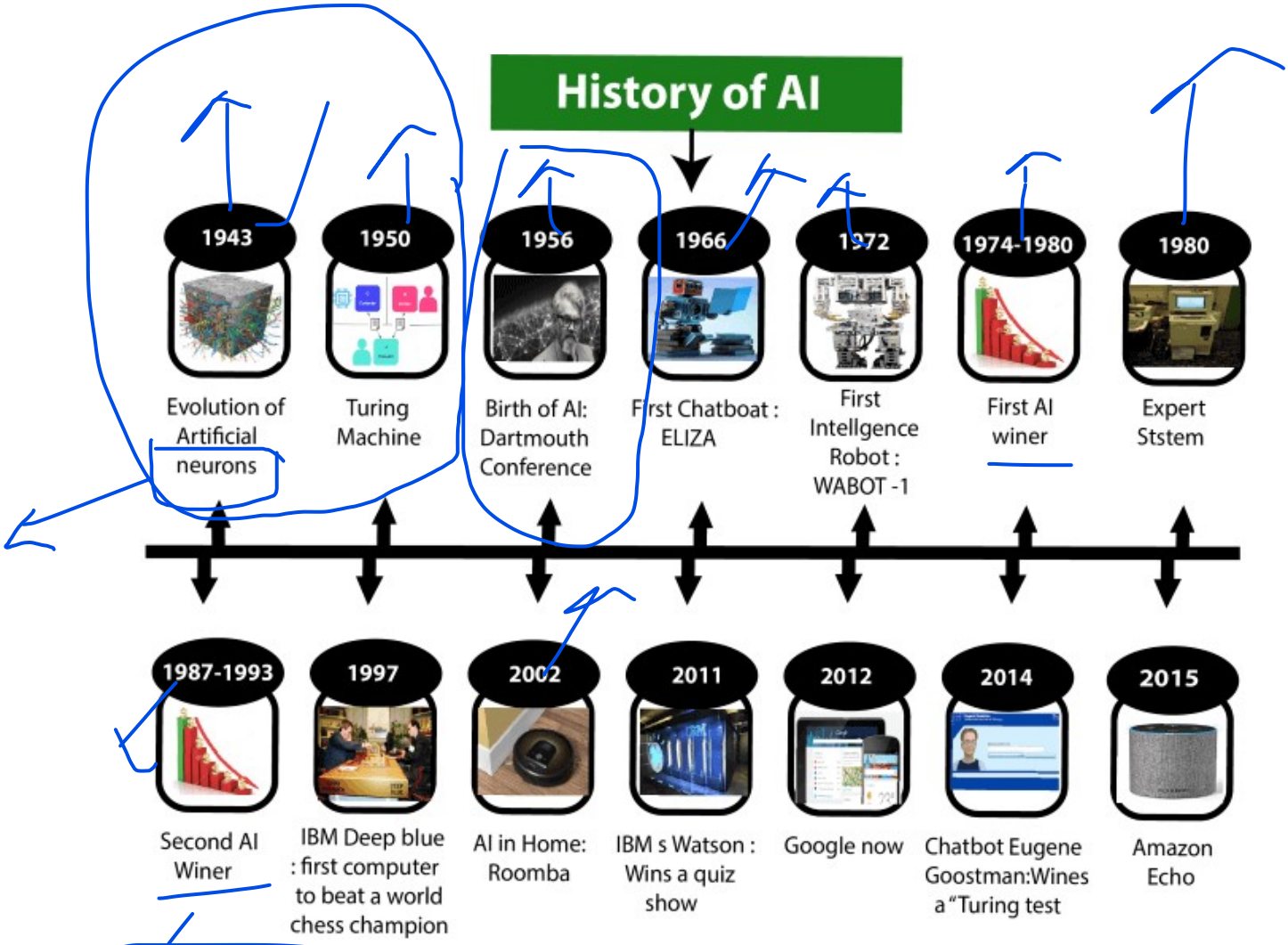
- AI is providing a competitive edge to the e-commerce industry, and it is becoming more demanding in the e-commerce business. AI is helping shoppers to discover associated products with recommended size, color, or even brand.

13. AI in education:

- AI can automate grading so that the tutor can have more time to teach. AI chatbot can communicate with students as a teaching assistant.
- AI in the future can be work as a personal virtual tutor for students, which will be accessible easily at any time and any place.

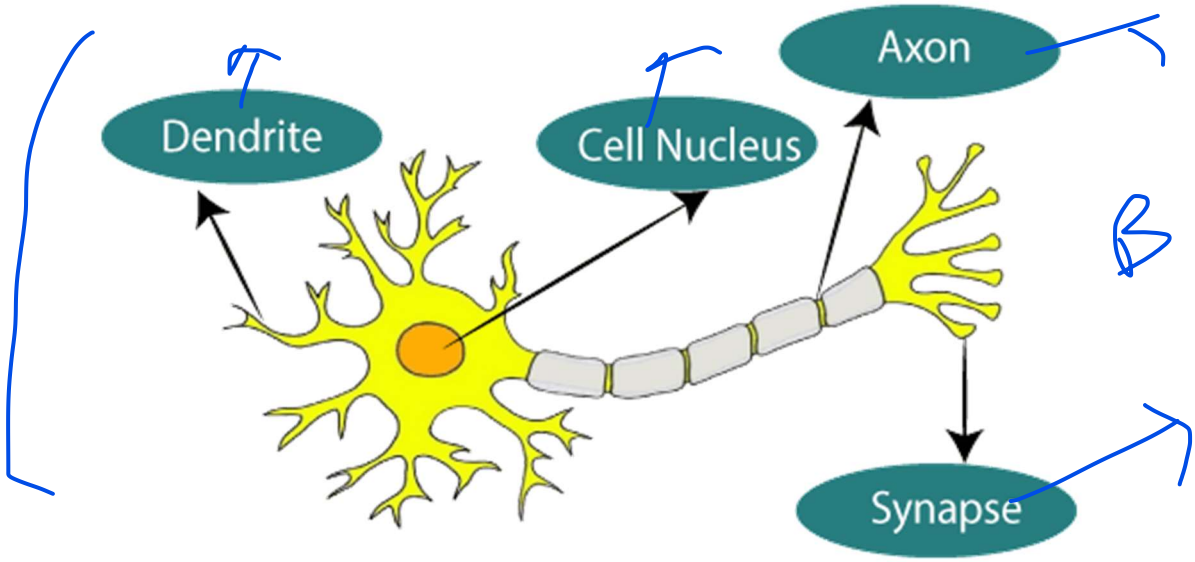
History of Artificial Intelligence

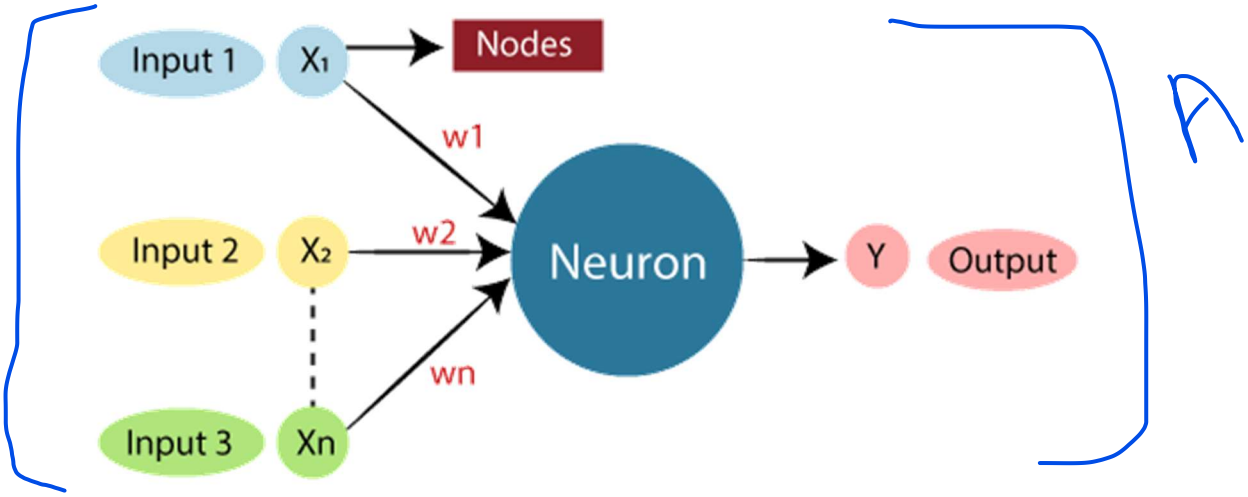
Artificial Intelligence is not a new word and not a new technology for researchers. This technology is much older than you would imagine. Even there are the myths of Mechanical men in Ancient Greek and Egyptian Myths. Following are some milestones in the history of AI which defines the journey from the AI generation to till date development.



Maturation of Artificial Intelligence (1943-1952)

- **Year 1943:** The first work which is now recognized as AI was done by Warren McCulloch and Walter pits in 1943. They proposed a model of **artificial neurons**.
- **Year 1949:** Donald Hebb demonstrated an updating rule for modifying the connection strength between neurons. His rule is now called **Hebbian learning**.





Dendrites from Biological Neural Network represent inputs in Artificial Neural Networks, cell nucleus represents Nodes, synapse represents Weights, and Axon represents Output.

- **Year 1950:** The Alan Turing who was an English mathematician and pioneered Machine learning in 1950. Alan Turing publishes **"Computing Machinery and Intelligence"** in which he proposed a test. The test can check the machine's ability to exhibit intelligent behavior equivalent to human intelligence, called a **Turing test**.

The birth of Artificial Intelligence (1952-1956)

- **Year 1955:** An Allen Newell and Herbert A. Simon created the "first artificial intelligence program "Which was named as **"Logic Theorist"**. This program had proved 38 of 52 Mathematics theorems, and find new and more elegant proofs for some theorems.
- **Year 1956:** The word "Artificial Intelligence" first adopted by American Computer scientist John McCarthy at the Dartmouth Conference. For the first time, AI coined as an academic field.

At that time high-level computer languages such as FORTRAN, LISP, or COBOL were invented. And the enthusiasm for AI was very high at that time.

The golden years-Early enthusiasm (1956-1974)

- **Year 1966:** The researchers emphasized developing algorithms which can solve mathematical problems. Joseph Weizenbaum created the first chatbot in 1966, which was named as ELIZA.
- **Year 1972:** The first intelligent humanoid robot was built in Japan which was named as WABOT-1.

The first AI winter (1974-1980)

- The duration between years 1974 to 1980 was the first AI winter duration. AI winter refers to the time period where computer scientist dealt with a severe shortage of funding from government for AI researches.
- During AI winters, an interest of publicity on artificial intelligence was decreased.

A boom of AI (1980-1987)

- **Year 1980:** After AI winter duration, AI came back with "Expert System". Expert systems were programmed that emulate the decision-making ability of a human expert.
- In the Year 1980, the first national conference of the American Association of Artificial Intelligence **was held at Stanford University**.

The second AI winter (1987-1993)

- The duration between the years 1987 to 1993 was the second AI Winter duration.
- Again Investors and government stopped in funding for AI research as due to high cost but not efficient result. The expert system such as XCON was very cost effective.

The emergence of intelligent agents (1993-2011)

- **Year 1997:** In the year 1997, IBM Deep Blue beats world chess champion, Gary Kasparov, and became the first computer to beat a world chess champion.
- **Year 2002:** for the first time, AI entered the home in the form of Roomba, a vacuum cleaner.
- **Year 2006:** AI came in the Business world till the year 2006. Companies like Facebook, Twitter, and Netflix also started using AI.

Deep learning, big data and artificial general intelligence (2011-present)

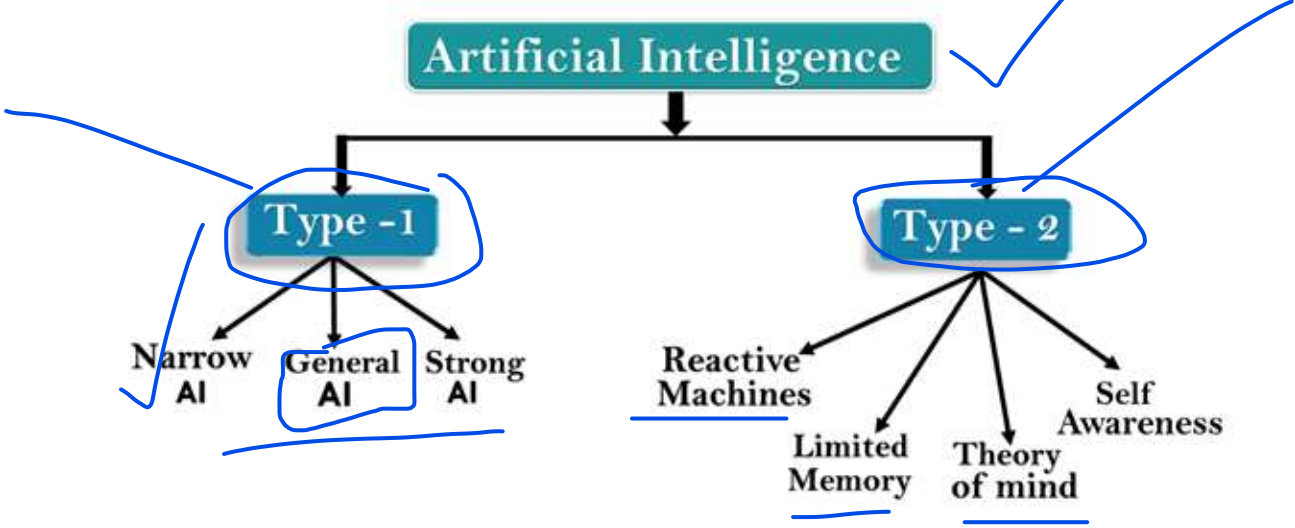
- **Year 2011:** In the year 2011, IBM's Watson won jeopardy, a quiz show, where it had to solve the complex questions as well as riddles. Watson had proved that it could understand natural language and can solve tricky questions quickly.
- **Year 2012:** Google has launched an Android app feature "Google now", which was able to provide information to the user as a prediction.
- **Year 2014:** In the year 2014, Chatbot "Eugene Goostman" won a competition in the infamous "Turing test."
- **Year 2018:** The "Project Debater" from IBM debated on complex topics with two master debaters and also performed extremely well.
- Google has demonstrated an AI program "Duplex" which was a virtual assistant and which had taken hairdresser appointment on call, and lady on other side didn't notice that she was talking with the machine.

Now AI has developed to a remarkable level. The concept of Deep learning, big data, and data science are now trending like a boom. Nowadays companies like Google, Facebook,

IBM, and Amazon are working with AI and creating amazing devices. The future of Artificial Intelligence is inspiring and will come with high intelligence.

Types of Artificial Intelligence:

Artificial Intelligence can be divided in various types, there are mainly two types of main categorization which are based on capabilities and based on functionally of AI. Following is flow diagram which explain the types of AI.



AI type-1: Based on Capabilities

1. Weak AI or Narrow AI:

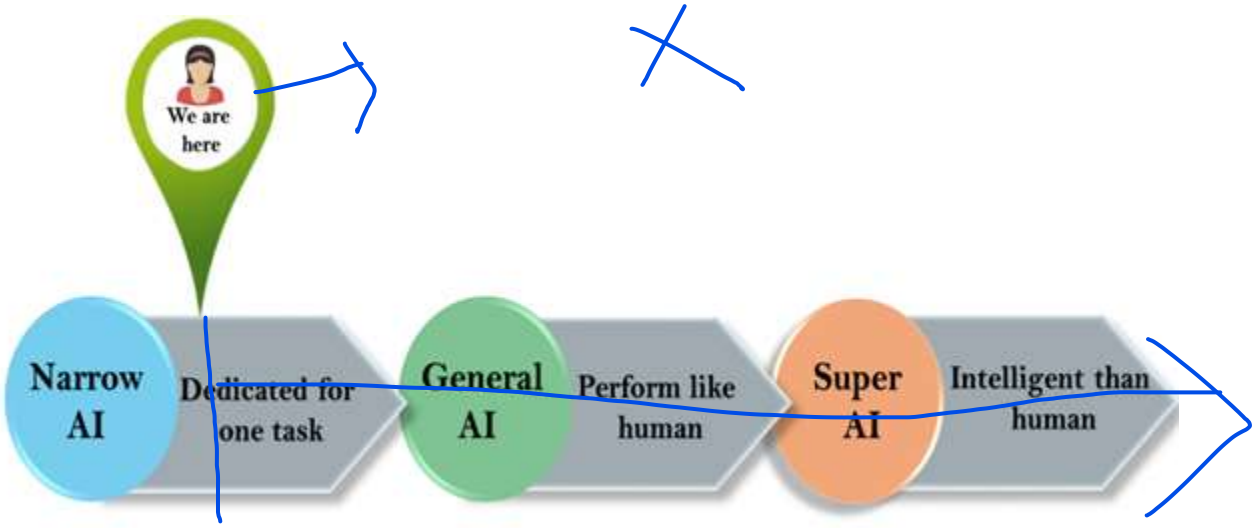
- Narrow AI is a type of AI which is able to perform a dedicated task with intelligence. The most common and currently available AI is Narrow AI in the world of Artificial Intelligence.
- Narrow AI cannot perform beyond its field or limitations, as it is only trained for one specific task. Hence it is also termed as weak AI. Narrow AI can fail in unpredictable ways if it goes beyond its limits.
- Apple Siris a good example of Narrow AI, but it operates with a limited pre-defined range of functions.
- IBM's Watson supercomputer also comes under Narrow AI, as it uses an Expert system approach combined with Machine learning and natural language processing.
- Some Examples of Narrow AI are playing chess, purchasing suggestions on e-commerce site, self-driving cars, speech recognition, and image recognition.

2. General AI:

- General AI is a type of intelligence which could perform any intellectual task with efficiency like a human.
- The idea behind the general AI to make such a system which could be smarter and think like a human by its own.
- Currently, there is no such system exist which could come under general AI and can perform any task as perfect as a human.
- The worldwide researchers are now focused on developing machines with General AI.
- As systems with general AI are still under research, and it will take lots of efforts and time to develop such systems.

3. Super AI:

- Super AI is a level of Intelligence of Systems at which machines could surpass human intelligence, and can perform any task better than human with cognitive properties. It is an outcome of general AI.
- Some key characteristics of strong AI include capability include the ability to think, to reason, solve the puzzle, make judgments, plan, learn, and communicate by its own.
- Super AI is still a hypothetical concept of Artificial Intelligence. Development of such systems in real is still world changing task.



Artificial Intelligence type-2: Based on functionality

1. Reactive Machines

- Purely reactive machines are the most basic types of Artificial Intelligence.
- Such AI systems do not store memories or past experiences for future actions.
- These machines only focus on current scenarios and react on it as per possible best action.
- IBM's Deep Blue system(Deep Blue was a chess-playing expert system run on a unique purpose-built IBM supercomputer. It was the first computer to win a game, and the first to win a match, against a reigning world champion under regular time control) is an example of reactive machines.
- Google's AlphaGo (AlphaGo is a computer program that plays the board game Go. It was developed by the London-based DeepMind Technologies, an acquired subsidiary of Google (now Alphabet Inc.).) is also an example of reactive machines.

2. Limited Memory

- Limited memory machines can store past experiences or some data for a short period of time.
- These machines can use stored data for a limited time period only.
- Self-driving cars are one of the best examples of Limited Memory systems. These cars can store recent speed of nearby cars, the distance of other cars, speed limit, and other information to navigate the road.

3. Theory of Mind

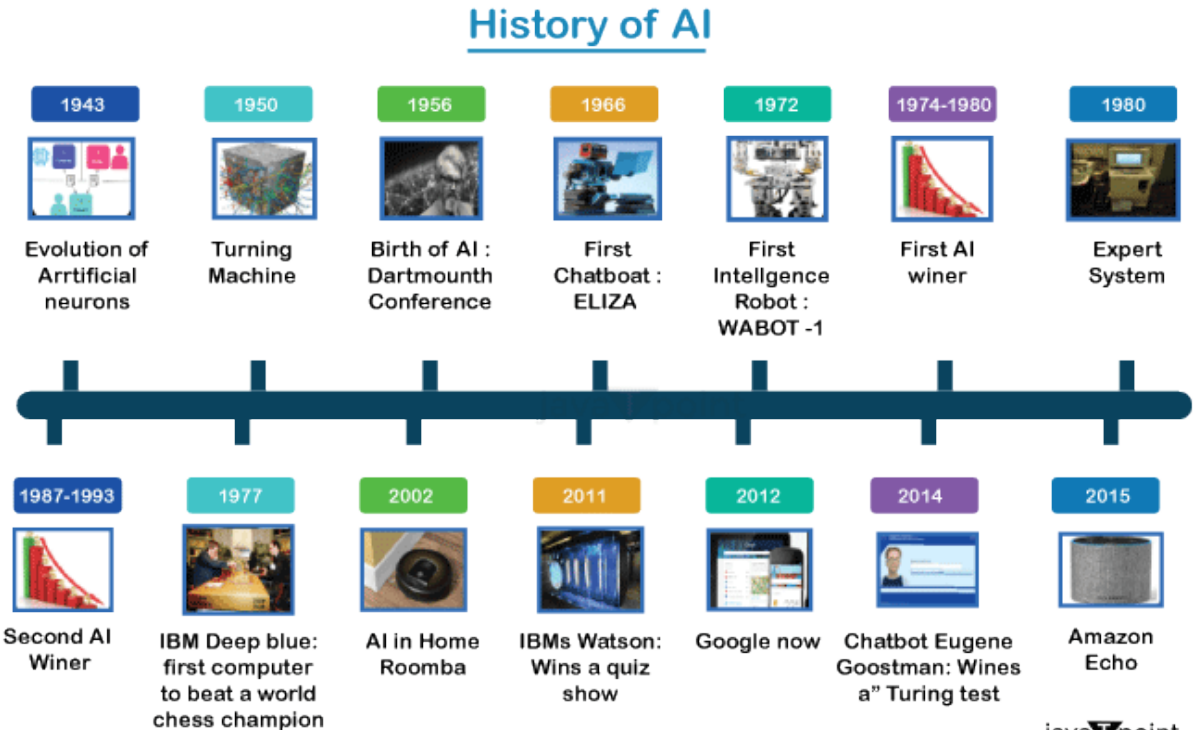
- Theory of Mind AI should understand the human emotions, people, beliefs, and be able to interact socially like humans.
- This type of AI machines are still not developed, but researchers are making lots of efforts and improvement for developing such AI machines.

4. Self-Awareness

- Self-awareness AI is the future of Artificial Intelligence. These machines will be super intelligent, and will have their own consciousness, sentiments, and self-awareness.
- These machines will be smarter than human mind.
- Self-Awareness AI does not exist in reality still and it is a hypothetical concept.

UNIT 1

Scientists have been working on artificial intelligence since the middle of the last century. Their goal: To develop machines that learn and think like humans. Here is an overview of the key learnings and technological milestones they have reached.



1936: Turing machine

The British mathematician Alan Turing applies his theories to prove that a computing machine — known as a 'Turing machine' — would be capable of executing cognitive processes, provided they could be broken down into multiple, individual steps and represented by an algorithm. In doing so, he lays the foundation for what we call artificial intelligence today.

1966: Birth of the first chatbot

The German-American computer scientist Joseph Weizenbaum of the Massachusetts Institute of Technology invents a computer program that communicates with humans. 'ELIZA' uses scripts to simulate various conversation partners such as a psychotherapist. Weizenbaum is surprised at the simplicity of the means required for ELIZA to create the illusion of a human conversation partner.

1972: AI enters the medical field

With 'MYCIN', artificial intelligence finds its way into medical practices: The expert system developed by Ted Shortliffe at Stanford University is used for the treatment of illnesses. Expert systems are computer programs that bundle the knowledge for a

specialist field using formulas, rules, and a knowledge database. They are used for diagnosis and treatment support in medicine.

1986: 'NETtalk' speaks
NETtalk is able to read words and pronounce them correctly, and can apply what it has learned to words it does not know. It is one of the early artificial neural networks — programs that are supplied with large datasets and are able to draw their own conclusions on this basis. Their structure and function are thereby similar to those of the human brain.

1997: Computer beats world chess champion
The AI chess computer 'Deep Blue' from IBM defeats the incumbent chess world champion Garry Kasparov in a tournament. This is considered a historic success in an area previously dominated by humans.

2011: AI enters everyday life

Technology leaps in the hardware and software fields pave the way for artificial intelligence to enter everyday life. Powerful processors and graphics cards in computers, smartphones, and tablets give regular consumers access to AI programs. Digital assistants in particular enjoy great popularity: Apple's 'Siri' comes to the market in 2011, Microsoft introduces the 'Cortana' software in 2014, and Amazon presents Amazon Echo with the voice service 'Alexa' in 2015.

2011: AI 'Watson' wins quiz show

The computer program 'Watson' competes in a U.S. television quiz show in the form of an animated on-screen symbol and wins against the human players. In doing so, Watson proves that it understands natural language and is able to answer difficult questions quickly.

20xx: The near future is intelligent
It needs to become more reliable and secure against manipulation before it can be used in sensitive areas, such as autonomous driving or medicine. Another goal is for AI systems to learn to explain their decisions so that humans can comprehend them and better research how AI thinks.

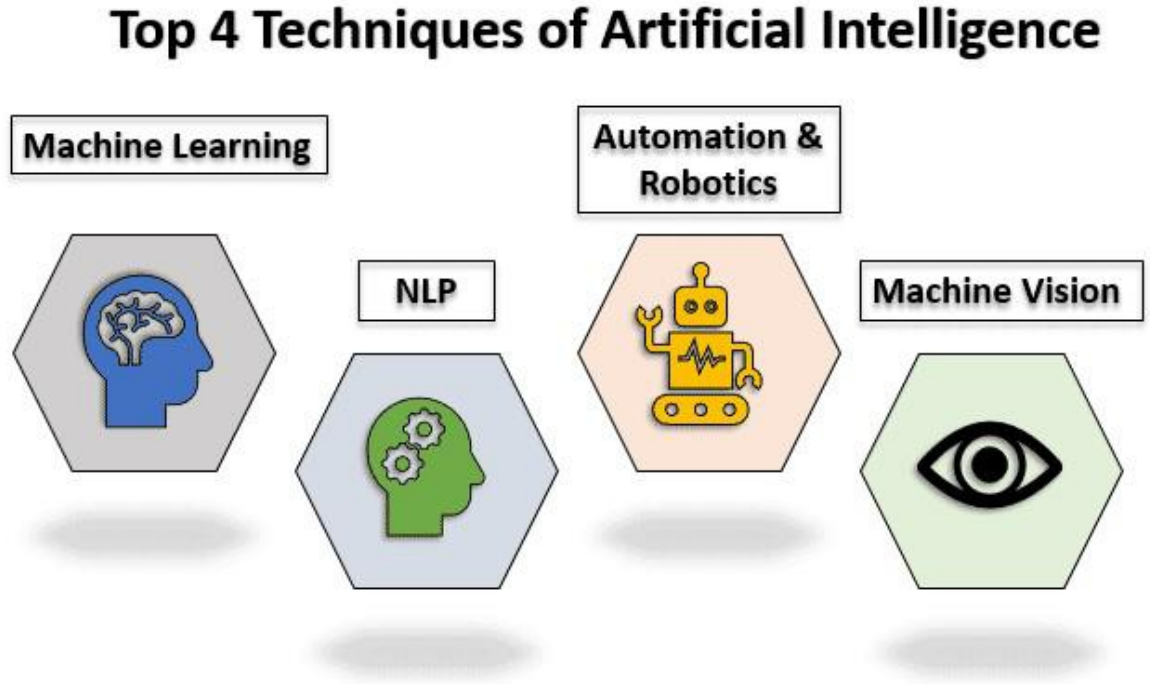
Some common AI applications include:

- Virtual assistants like Siri and Alexa
- Recommendation systems used in e-commerce platforms
- Fraud detection in financial institutions
- Autonomous vehicles

- NLP for chatbots and customer service
- Image and facial recognition in security systems
- Medical diagnosis and healthcare systems

What is an AI Technique?

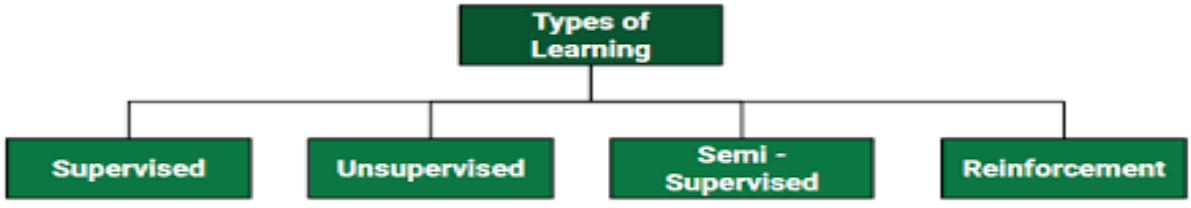
Artificial Intelligence (AI) refers to developing computer systems for performing tasks requiring human intelligence. These systems assess large amounts of data to identify patterns and make logical decisions based on the collected information. The ultimate goal of AI is to create machines to carry out diverse tasks.



Artificial Intelligence techniques refer to a set of methods and algorithms used to develop intelligent systems that can perform tasks requiring human-like intelligence. Some of the widely used ones are:

- **Machine Learning.**
- **Natural Language Processing.**
- **Computer Vision.**
- **Deep Learning**
- **Robotics.**

<u>Machine Learning:</u>



1. Unsupervised machine learning -AI systems analyse unlabelled data, where no predefined outcomes are provided. The objective is to uncover inherent structures or patterns within the data without any prior knowledge. For instance, it can group similar customer behaviour data to identify customer segments for targeted marketing strategies.

2. Supervised learning – A combination of an input data set and the intended output is inferred from the training data. AI systems learn from a labelled dataset, where each data point is associated with a known outcome. For instance, it enables email spam filters to distinguish between spam and legitimate emails based on learned patterns.

3. Semi-supervised learning – It is a method that uses a small amount of labelled data and a large amount of unlabelled data to train a model. The goal of semi-supervised learning is to learn a function that can accurately predict the output variable based on the input variables, similar to supervised learning. However, unlike supervised learning, the algorithm is trained on a dataset that contains both labelled and unlabelled data.

4. Reinforcement learning – In RL, the data is accumulated from machine learning systems that use a trial-and-error method to learn from outcomes and decide which action to take next. After each action, the algorithm receives feedback that helps it determine whether the choice it made was correct, neutral or incorrect. It performs actions with the aim of maximizing rewards, or in other words, it is learning by doing in order to achieve the best outcomes.

Natural Language Processing:

Natural Language Processing involves programming computers to process human languages to facilitate interactions between humans and computers.

However, the nature of human languages makes Natural Language Processing difficult because of the rules involved in passing information using natural language. NLP leverages algorithms to recognize and abstract the rules of natural languages, converting unstructured human language data into a computer-understandable format.

Computer Vision:

Computer Vision equips machines with the ability to interpret visual information from the world. This technique has revolutionized industries like healthcare, automotive, and robotics, enabling tasks such as facial recognition, object detection, and autonomous driving. The extent to which it can discriminate between objects is an essential component of machine vision.

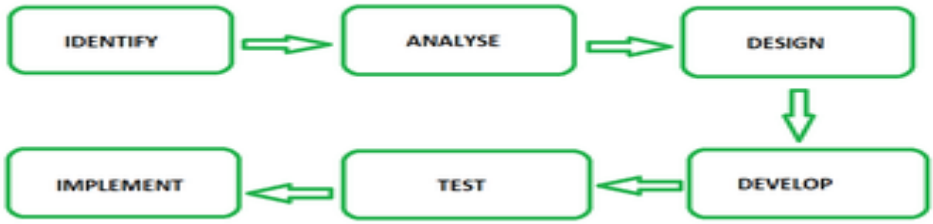
Sensitivity in computer vision is an AI application's ability to pick out small details in visual information. A low-sensitivity system may not pick up subtle clues in images or fail to work well in low lighting. However, high sensitivity might be able to look at an

image's fine details and pick up on information other systems might miss. A common example is Surveillance systems.
Resolution is the level of detail a computer vision system can capture and process. High-resolution images are vital for correctly identifying an image's detail.

Robotics & Automation

Automation aims to enable machines to perform boring, repetitive jobs, increasing productivity and delivering more effective, efficient, and affordable results. To automate processes, many businesses employ machine learning, artificial neural, and graphs.

By leveraging the CAPTCHA technique, this automation can avoid fraud problems during online payments.
Robotic process automation is designed to carry out high-volume, repetitive jobs while being capable of adapting to changing conditions.



Deep Learning:

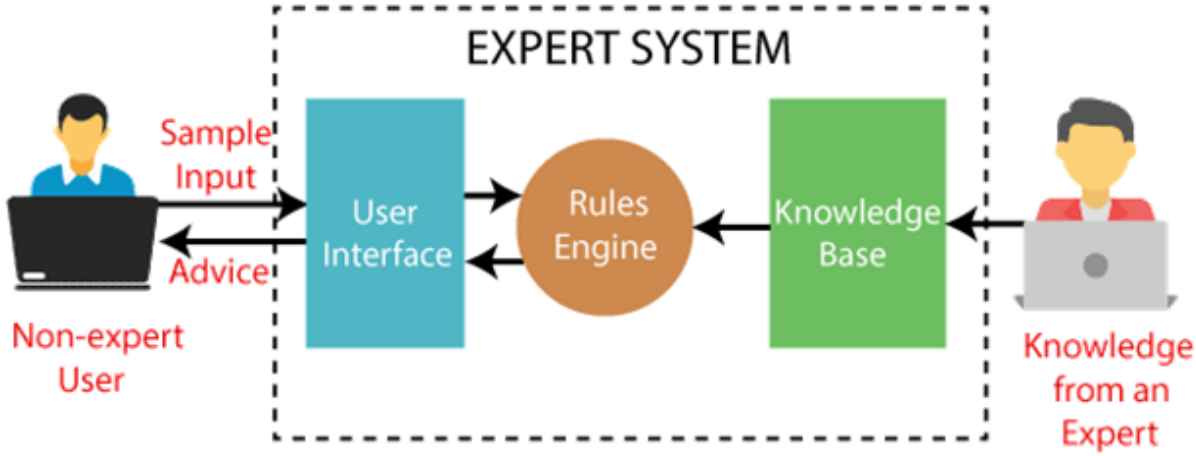
Deep learning is the branch of machine learning which is based on artificial neural network architecture. An artificial neural network or ANN uses layers of interconnected nodes called neurons that work together to process and learn from the input data.

What is an Expert System?

An expert system is a computer program that is designed to solve complex problems and to provide decision-making ability like a human expert. It performs this by extracting knowledge from its knowledge base using the reasoning and inference rules according to the user queries.

The performance of an expert system is based on the expert's knowledge stored in its knowledge base. The more knowledge stored in the KB, the more that system improves its performance. One of the common examples of an ES is a suggestion of spelling errors while typing in the Google search box.

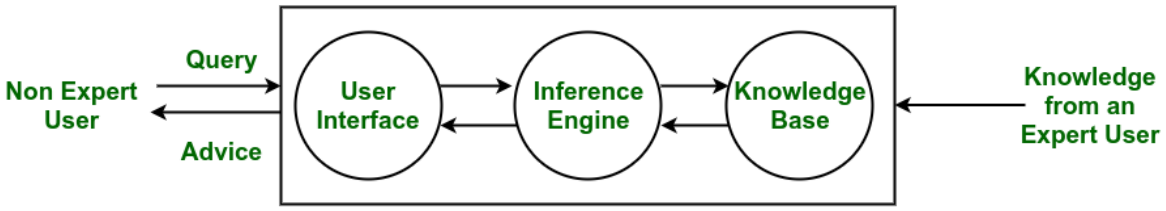
Below is the block diagram that represents the working of an expert system.



Below are some popular examples of the Expert System:

- **DENDRAL:** It was an artificial intelligence project that was made as a chemical analysis expert system. It was used in organic chemistry to detect unknown organic molecules with the help of their mass spectra and knowledge base of chemistry.
- **MYCIN:** It was one of the earliest backward chaining expert systems that was designed to find the bacteria causing infections like bacteraemia and meningitis. It was also used for the recommendation of antibiotics and the diagnosis of blood clotting diseases.
- **PXDES:** It is an expert system that is used to determine the type and level of lung cancer. To determine the disease, it takes a picture from the upper body, which looks like the shadow. This shadow identifies the type and degree of harm.
- **CaDeT:** The CaDet expert system is a diagnostic support system that can detect cancer at early stages.

Components of an Expert System :



- **Knowledge Base –** The knowledge base represents facts and rules. It consists of knowledge in a particular domain as well as rules to solve a problem, procedures and intrinsic data relevant to the domain.

- **Inference Engine –**
  The function of the inference engine is to fetch the relevant knowledge from the knowledge base, interpret it and to find a solution relevant to the user's problem. The inference engine acquires the rules from its knowledge base and applies them to the known facts to infer new facts. Inference engines can also include an explanation and debugging abilities.

- **Knowledge Acquisition and Learning Module –**
  The function of this component is to allow the expert system to acquire more and more knowledge from various sources and store it in the knowledge base.

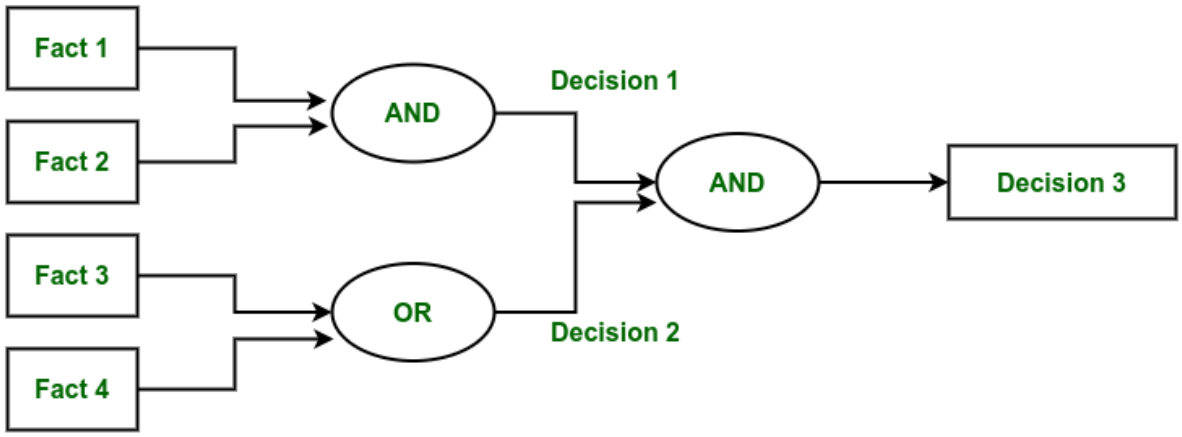
- **User Interface –**
  This module makes it possible for a non-expert user to interact with the expert system and find a solution to the problem.

- **Explanation Module –**
  This module helps the expert system to give the user an explanation about how the expert system reached a particular conclusion.

The Inference Engine generally uses two strategies for acquiring knowledge from the Knowledge Base, namely –

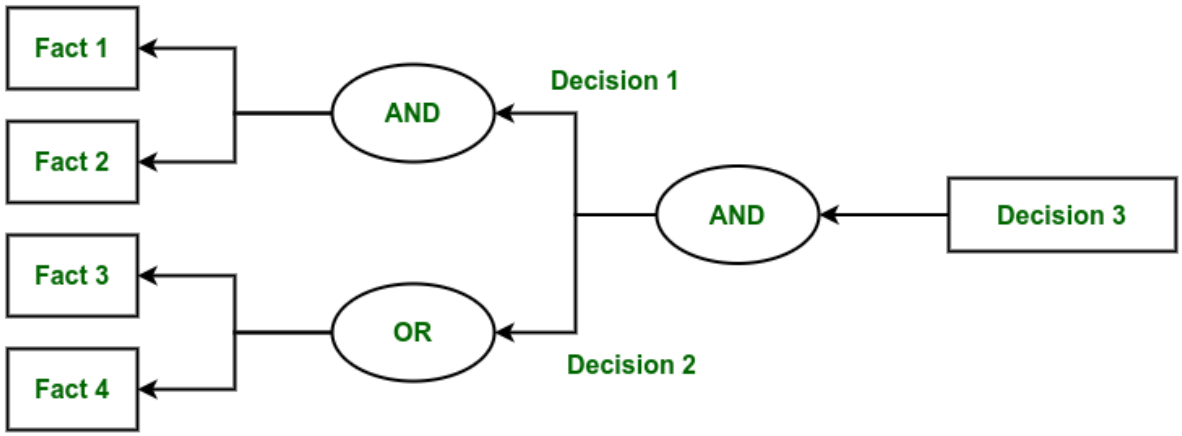
- Forward Chaining
- Backward Chaining

Forward Chaining –
Forward Chaining is a strategic process used by the Expert System to answer the questions – What will happen next. This strategy is mostly used for managing tasks like creating a conclusion, result or effect. Example – prediction or share market movement status.



Forward Chaining

Backward Chaining –
Backward Chaining is a strategy used by the Expert System to answer the questions – Why this has happened. This strategy is mostly used to find out the root cause or reason behind it, considering what has already happened. Example – diagnosis of stomach pain, blood cancer or dengue, etc.



Backward Chaining

Difference between Forwarding Chaining and Backward Chaining:

	Forward Chaining	Backward Chaining
1.	When based on available data a decision is taken then the process is called as Forward chaining.	Backward chaining starts from the goal and works backward to determine what facts must be asserted so that the goal can be achieved.
2.	Forward chaining is known as data-driven technique because we reaches to the goal using the available data.	Backward chaining is known as goal-driven technique because we start from the goal and reaches the initial state in order to extract the facts.
3.	It is a bottom-up approach.	It is a top-down approach.
4.	It applies the Breadth-First Strategy.	It applies the Depth-First Strategy.
5.	Its goal is to get the conclusion.	Its goal is to get the possible facts or the required data.

6.	**Slow as it has to use all the rules.**	**Fast as it has to use only a few rules.**
7.	**It operates in forward direction i.e it works from initial state to final decision.**	**It operates in backward direction i.e it works from goal to reach initial state.**
8.	**Forward chaining is used for the planning, monitoring, control, and interpretation application.**	**It is used in automated inference engines, theorem proofs, proof assistants and other artificial intelligence applications.**

Characteristics of Expert System

- **High Performance:** The expert system provides high performance for solving any type of complex problem of a specific domain with high efficiency and accuracy.
- **Understandable:** It responds in a way that can be easily understandable by the user. It can take input in human language and provides the output in the same way.
- **Reliable:** It is much reliable for generating an efficient and accurate output.
- **Highly responsive:** ES provides the result for any complex query within a very short period of time.

Advantages :

- Low accessibility cost.
- Fast response.
- Not affected by emotions, unlike humans.
- Low error rate.
- Capable of explaining how they reached a solution.

Disadvantages :

- The expert system has no emotions.
- Common sense is the main issue of the expert system.
- It is developed for a specific domain.
- It needs to be updated manually. It does not learn itself.
- Not capable to explain the logic behind the decision.

Applications

The application of an expert system can be found in almost all areas of business or government. They include areas such as –

- Different types of medical diagnosis like internal medicine, blood diseases and show on.
- Diagnosis of the complex electronic and electromechanical system.
- Diagnosis of a software development project.
- Planning experiment in biology, chemistry and molecular genetics.
- Forecasting crop damage.
- Diagnosis of the diesel-electric locomotive system.
- Identification of chemical compound structure.
- Scheduling of customer order, computer resources and various manufacturing task.
- Assessment of geologic structure from dip meter logs.
- Assessment of space structure through satellite and robot.
- The design of VLSI system.
- Teaching students specialize task.
- Assessment of log including civil case evaluation, product liability etc.

What is an AI agent?

An AI agent is a software that performs tasks on behalf of a user. They can automate processes, make decisions, and intelligently interact with their environment.

“AI agents are like magic,” said Patrick Hamelin, software engineer lead at Botpress. “They’re these magical entities that go beyond typical chatbots.”

AI agents are entities designed to perceive their environment and take actions in order to achieve specific goals. These agents can be software-based or physical entities.

They perceive their environment through sensors, process the information using algorithms or models, and then take actions using actuators or other means.

What’s the difference between an AI agent and an AI chatbot?

AI agents and chatbots differ in their purpose and capability. Chatbots are designed to interact with humans, while agents are designed to complete autonomous tasks.

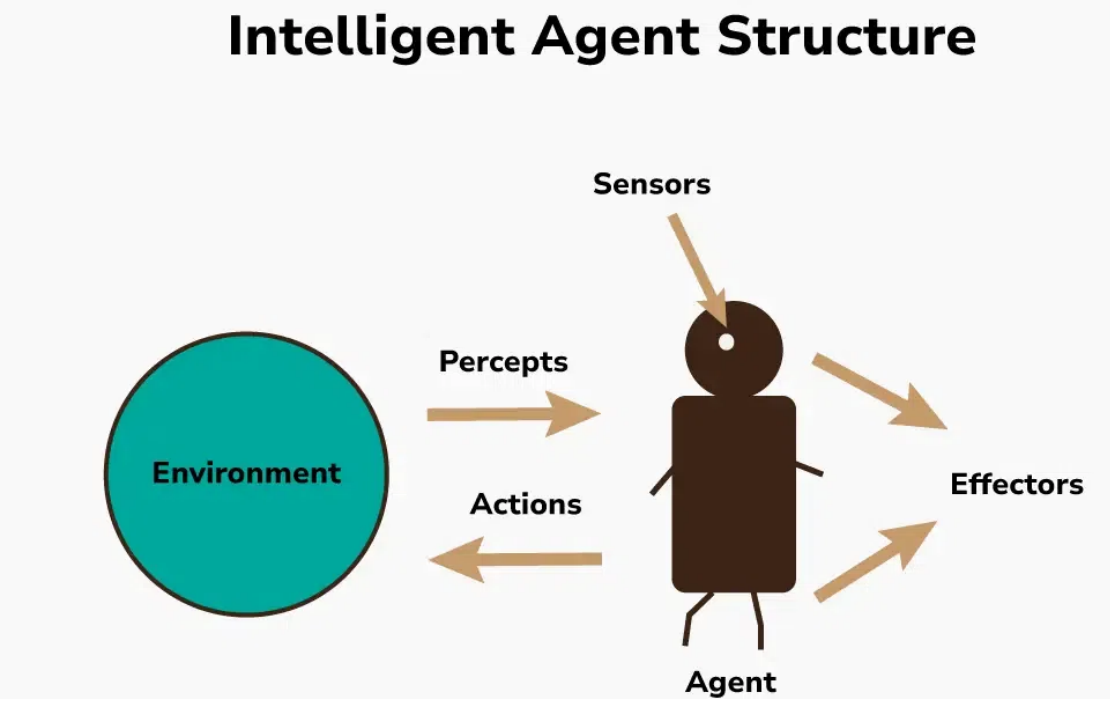
The biggest difference is their ability to take autonomous actions. Since AI chatbots are designed for conversation with humans, they’re not usually programmed to take autonomous action – their purpose is to directly assist a human.

AI agents, on the other hand, may not interact with a user at all. In some cases, they’ll receive a task from a developer and follow through on it independently, without interacting with another human.

How Intelligent Agent work Inside?

An agent's internal workings involve Agent program that run on computing device and process the data comes from the environment through its architecture. Let's discuss how an agent works from the inside using program and architecture:

Agent architecture



1. **Environment:** Environment is the area around the agent that it interacts with. An environment can be anything like a physical space, a room or a virtual space like a game world or the internet.

2. **Sensors:** Sensors are tools that AI agent uses to perceive their environment. They can be any physical like cameras, microphones, temperature sensors or a software sensor that read data from files.

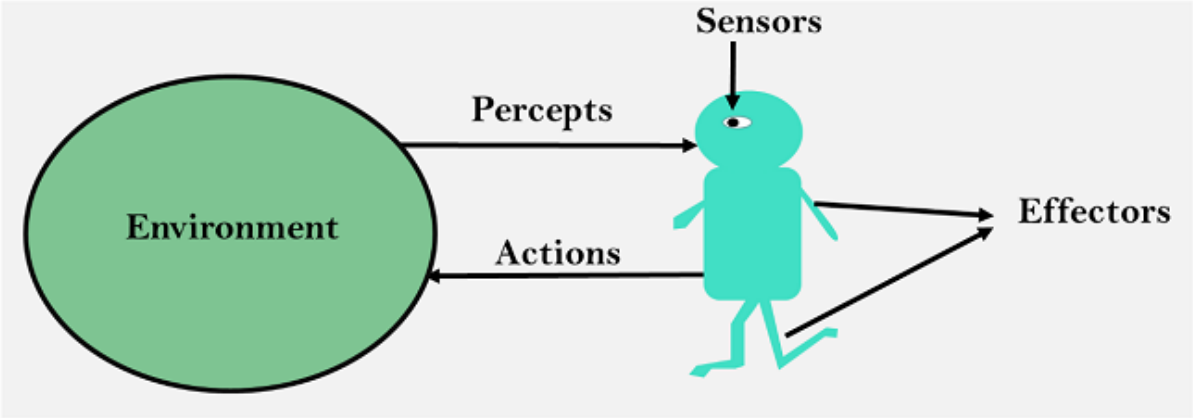
3. **Actuators:** Actuators are tools that AI agent uses to interact with their environment through some actions. They can be any physical actuators like wheels, motors, robotic hands, or computer screens or they can be software actuators that send messages.

4. **Effectors:** Effectors take instructions from decision making mechanism and translates them into actions and these actions are performed through actuators.

What is an Agent?

An agent can be anything that perceiveits environment through sensors and act upon that environment through actuators. An Agent runs in the cycle of **perceiving**, **thinking**, and **acting**. An agent can be:

- **Human-Agent:** A human agent has eyes, ears, and other organs which work for sensors and hand, legs, vocal tract work for actuators.
- **Robotic Agent:** A robotic agent can have cameras, infrared range finder, NLP for sensors and various motors for actuators.
- **Software Agent:** Software agent can have keystrokes, file contents as sensory input and act on those inputs and display output on the screen.



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.

Intelligent Agents:

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. A thermostat is an example 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 an AI agent must be a rational action.

Rational Agent:

A rational agent is an agent which has clear preference, models uncertainty, and acts in a way to maximize its performance measure with all possible actions.

A rational agent is said to perform the right things. AI is about creating rational agents to use for game theory and decision theory for various real-world scenarios.

Intelligent Agent vs. Rational Agent

	Intelligent Agent	**Rational Agent**
Definition	**An Intelligent Agent is a system that can perceive its environment and take actions to achieve a specific goal.**	**A Rational Agent is an Intelligent Agent that makes decisions based on logical reasoning and optimizes its behavior to achieve a specific goal.**
Perception	**An Intelligent Agent can perceive its environment through various sensors or inputs.**	**A Rational Agent's perception is based on the information available to it and logical reasoning.**
Decision-making	**It can make decisions based on a set of rules or a pre-defined algorithm.**	**It makes decisions based on logical reasoning and optimizes its behavior to achieve its goals.**
Learning	**An Intelligent Agent can learn from its environment and adapt its behavior.**	**A Rational Agent can also learn from its environment and adapt its behavior, but it does so based on logical reasoning.**
Autonomy	**It can operate independently of human intervention.**	**It can also operate independently of human intervention, but it does so based on logical reasoning.**
Goals	**An Intelligent Agent can be designed to achieve a specific**	**A Rational Agent has a specific goal and optimizes its behavior.**

	goal.	
Examples	**An Intelligent Agent can be a self-driving car, a virtual personal assistant, or a recommendation system.**	**A Rational Agent can be a financial advisor, a chess-playing program, or a logistics planner.**

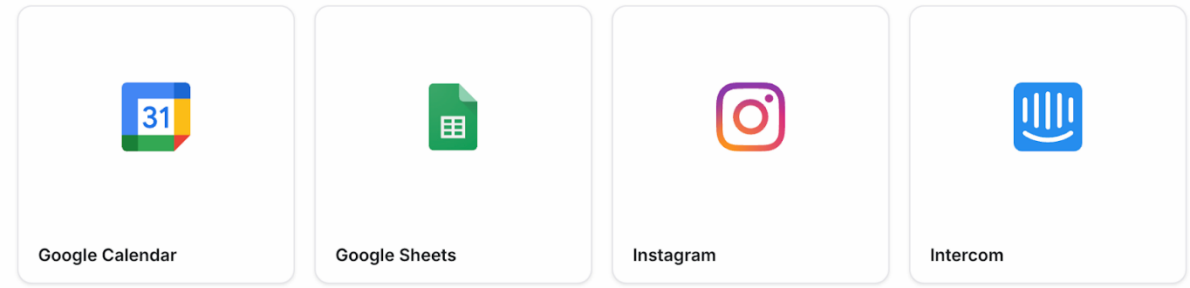
Applications of AI Agents

AI agents have a wide array of applications – they're beginning to make waves across numerous industries around the world. Here are a few of the most common:

Customer Service

Customer service chatbots are one of the most common types of AI agent deployment.

Because they can be plugged into company data, a business can use an AI agent to act as a customer assistant. They can provide access directly to the user's device anywhere in the world, including a webpage via their computer or different apps (like WhatsApp or Facebook Messenger).



These chatbots and virtual agents can point customers towards specific policies, give them an idea of what items might fulfill their needs, or even provide access to their account by resetting a password.

It's becoming expected for companies to offer customer service chatbots – most are powered by large language models and can complete specific tasks. The best ones are also able to take action on behalf of a business, like book a table or update a customer's record.

Autonomous Vehicles

One of the flashiest uses of AI agents are self-driving cars and drones. These vehicles can operate with limited human input, thanks to the power of AI agents.

AI agents are integral to their functioning – they perceive the car's environment and make informed decisions (like when it's safe to turn or when to slow down). They can identify when the car is approaching a stop sign or explore a new type of terrain by accounting for environmental inputs.

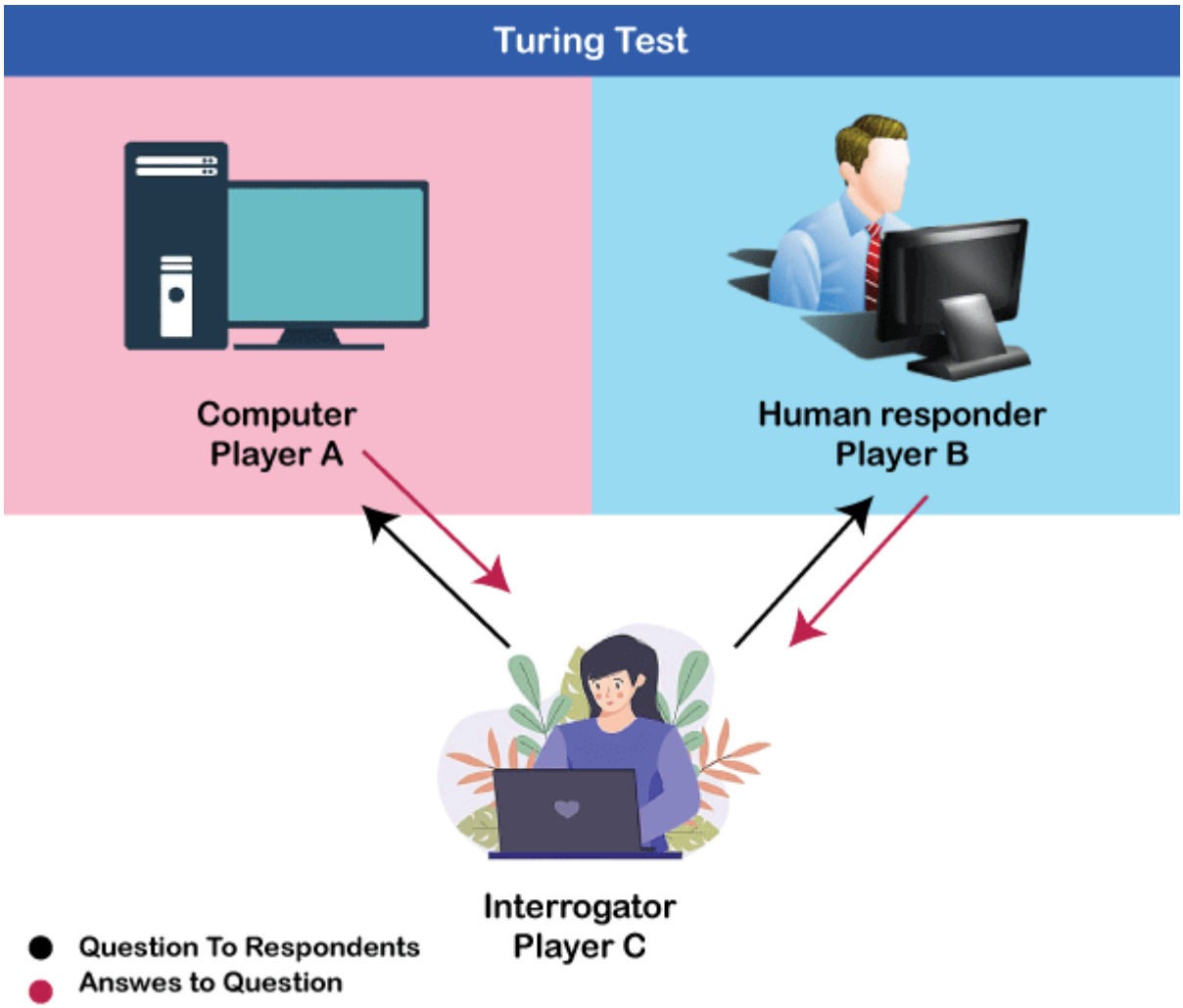
Virtual Assistants

Agents like Siri, Alexa, and Google Assistant use AI to understand natural language, assist with tasks, provide information, and control smart devices.

Turing Test in AI

In 1950, Alan Turing introduced a test to check whether a machine can think like a human or not, this test is known as the Turing Test. In this test, Turing proposed that the computer can be said to be an intelligent if it can mimic human response under specific conditions**.**

Turing Test was introduced by Turing in his 1950 paper, "Computing Machinery and Intelligence," which considered the question, "Can Machine think?"



The Turing test is based on a party game "Imitation game," with some modifications. This game involves three players in which one player is Computer, another player is human responder, and the third player is a human Interrogator, who is isolated from other two players and his job is to find that which player is machine among two of them.

Consider, Player A is a computer, Player B is human, and Player C is an interrogator. Interrogator is aware that one of them is machine, but he needs to identify this on the basis of questions and their responses.

The conversation between all players is via keyboard and screen so the result would not depend on the machine's ability to convert words as speech.

The test result does not depend on each correct answer, but only how closely its responses like a human answer. The computer is permitted to do everything possible to force a wrong identification by the interrogator.

Features required for a machine to pass the Turing test:

- **Natural language processing:** NLP is required to communicate with Interrogator in general human language like English.
- **Knowledge representation:** To store and retrieve information during the test.
- **Automated reasoning:** To use the previously stored information for answering the questions.
- **Machine learning:** To adapt new changes and can detect generalized patterns.
- **Vision (For total Turing test):** To recognize the interrogator actions and other objects during a test.
- **Motor Control (For total Turing test):** To act upon objects if requested.

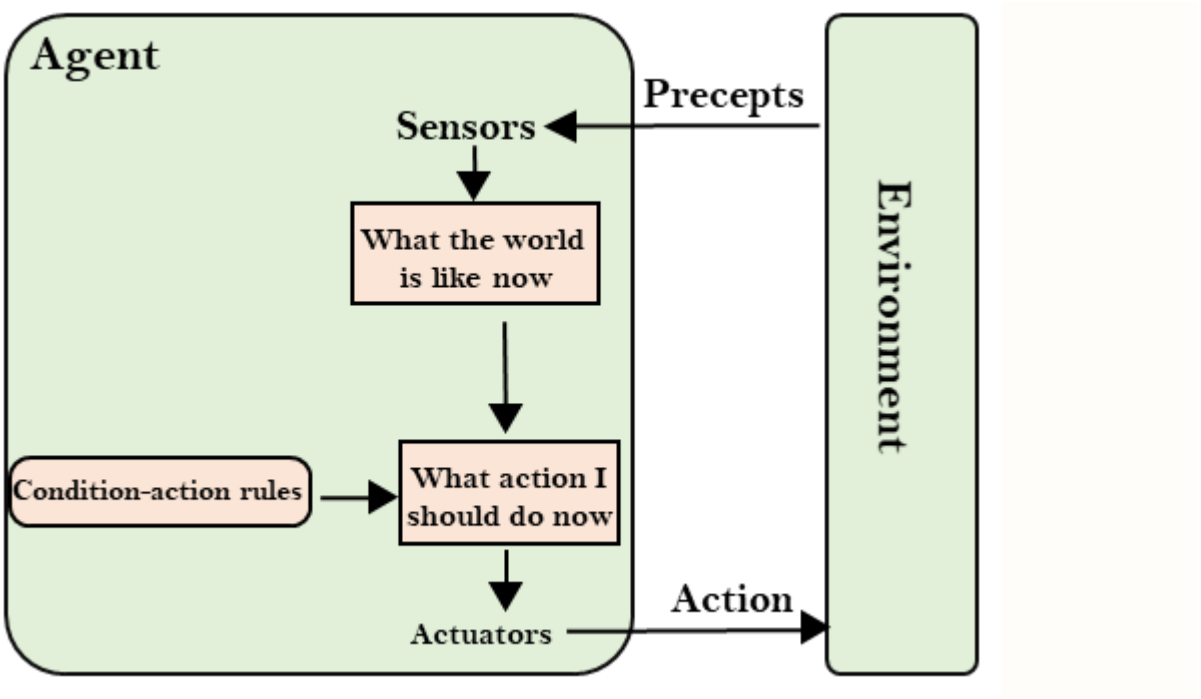
Types of AI agents

Based on their components, complexity, and real-world applications, here are the most common types of AI agents.

- Simple Reflex Agent
- Model-based reflex agent
- Goal-based agents
- Utility-based agent
- Learning agent

1. Simple Reflex agent:

- The Simple reflex agents are the simplest agents. These agents take decisions on the basis of the current percepts and ignore the rest of the percept history.
- These agents only succeed in the fully observable environment.
- The Simple reflex agent does not consider any part of percepts history during their decision and action process.
- The Simple reflex agent works on Condition-action rule, which means it maps the current state to action. Such as a Room Cleaner agent, it works only if there is dirt in the room.
- Problems for the simple reflex agent design approach:
    - They have very limited intelligence
    - They do not have knowledge of non-perceptual parts of the current state
    - Mostly too big to generate and to store.
    - Not adaptive to changes in the environment.



2. Model-based reflex agent

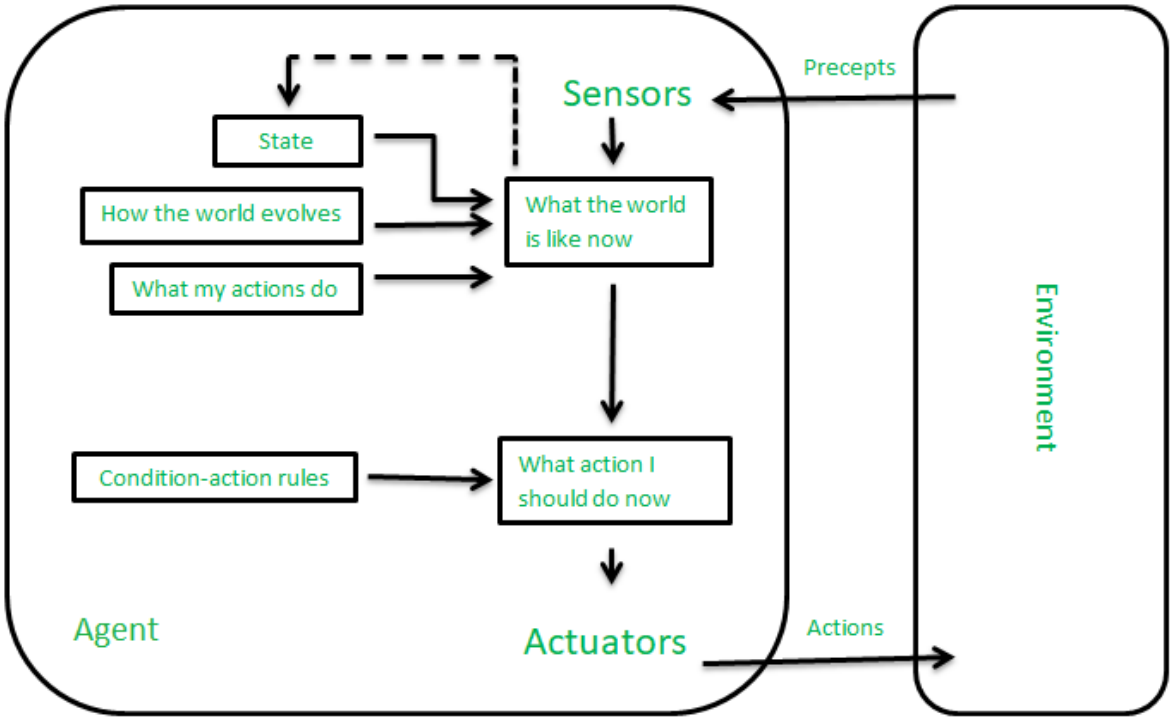
It works by finding a rule whose condition matches the current situation. A model-based agent can handle **partially observable environments** by the use of a model about the world.

The agent has to keep track of the **internal state** which is adjusted by each percept and that depends on the percept history. The current state is stored inside the agent which

maintains some kind of structure describing the part of the world which cannot be seen.

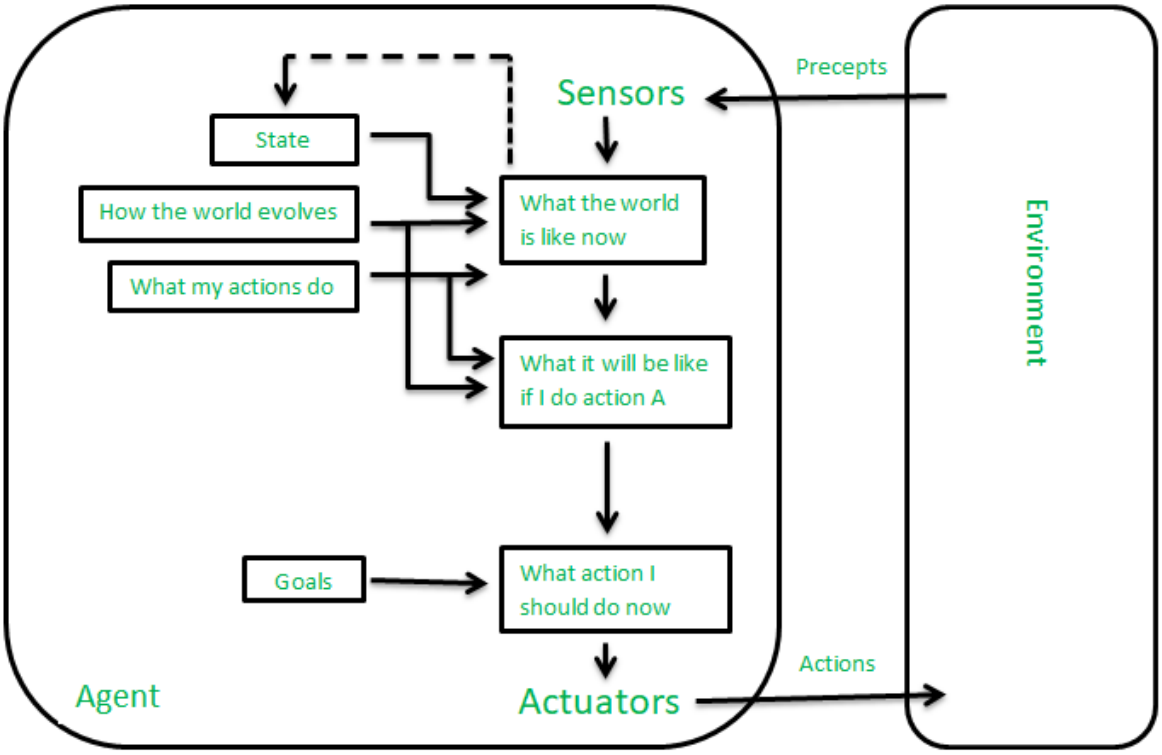
Updating the state requires information about:

- How the world evolves independently from the agent?
- How do the agent's actions affect the world?



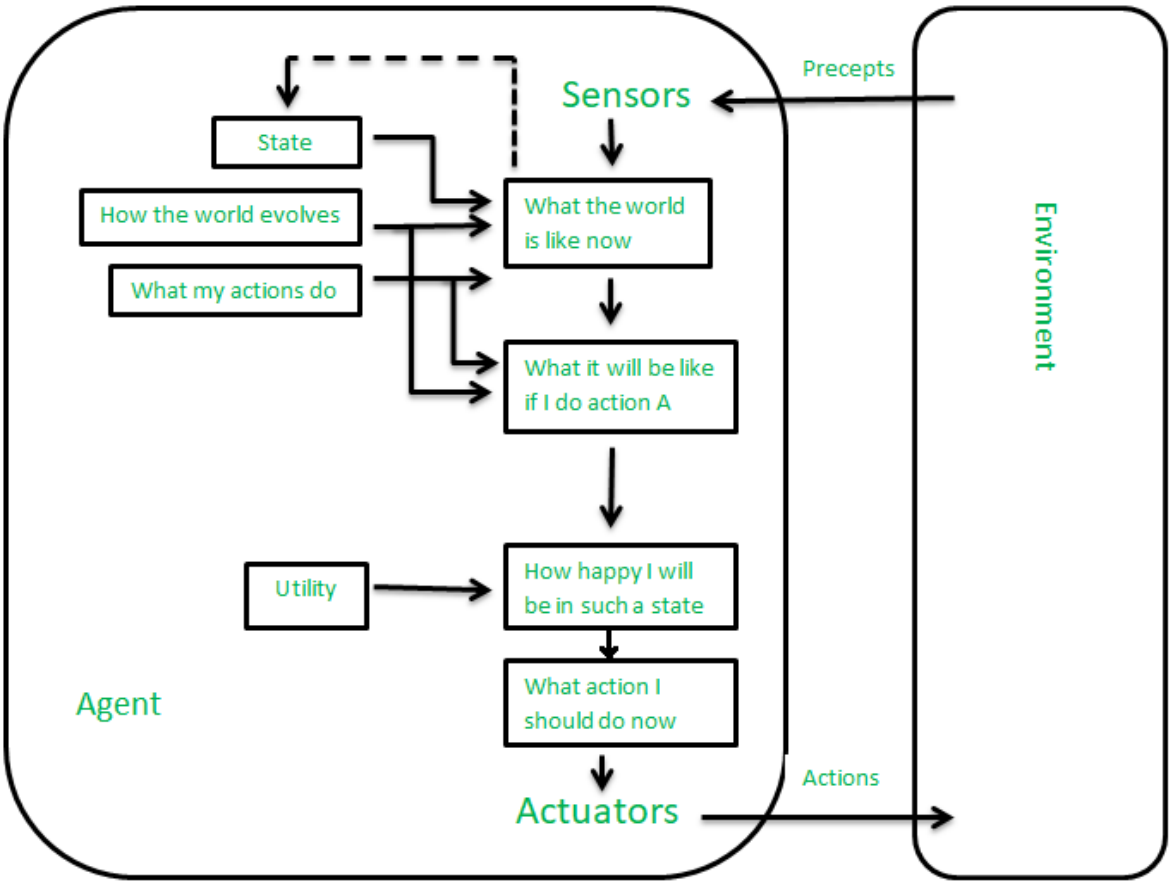
3. Goal-based agents

These kinds of agents take decisions based on how far they are currently from their **goal**(description of desirable situations). Their every action is intended to reduce their distance from the goal. This allows the agent a way to choose among multiple possibilities, selecting the one which reaches a goal state. They usually require search and planning. The goal-based agent's behavior can easily be changed.



4. Utility-based agents

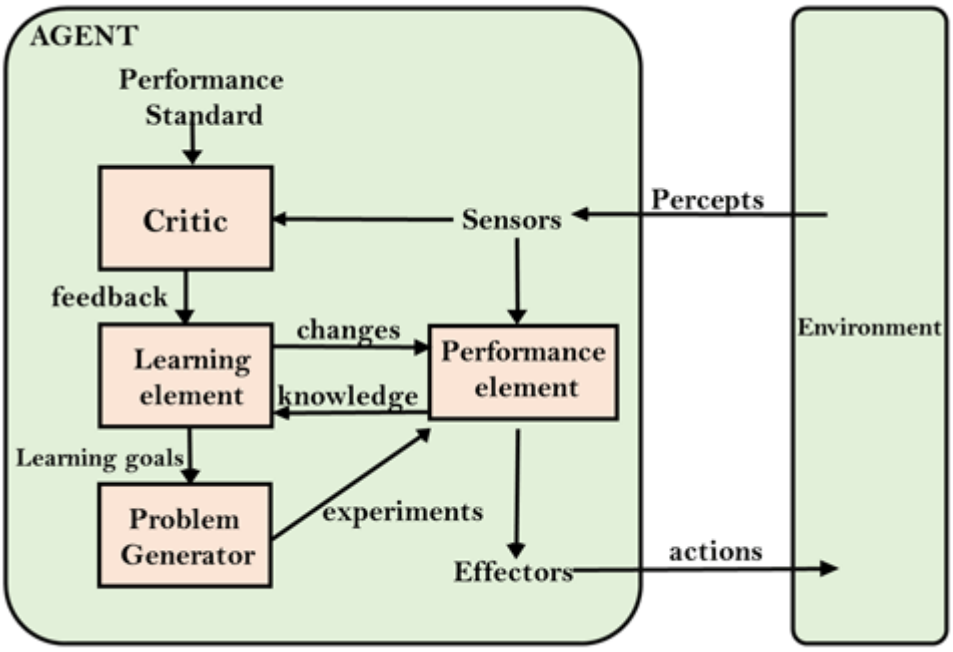
- These agents are similar to the goal-based agent but provide an extra component of utility measurement which makes them different by providing a measure of success at a given state.
- Utility-based agent act based not only goals but also the best way to achieve the goal.
- The Utility-based agent is useful when there are multiple possible alternatives, and an agent has to choose in order to perform the best action.
- Agent happiness should be taken into consideration. Utility describes how **"happy"** the agent is. Because of the uncertainty in the world, a utility agent chooses the action that maximizes the expected utility. A utility function maps a state onto a real number which describes the associated degree of happiness.



5. Learning Agents

A learning agent in AI is the type of agent that can learn from its past experiences or it has learning capabilities. It starts to act with basic knowledge and then is able to act and adapt automatically through learning. A learning agent has mainly four conceptual components, which are:

1. **Learning element:** It is responsible for making improvements by learning from the environment.
2. **Critic:** The learning element takes feedback from critics which describes how well the agent is doing with respect to a fixed performance standard.
3. **Performance element:** It is responsible for selecting external action.
4. **Problem Generator:** This component is responsible for suggesting actions that will lead to new and informative experiences.


AGENT
Performance Standard
Critic
Sensors
Percepts
feedback
Learning element
changes
knowledge
Performance element
Environment
Learning goals
Problem Generator
experiments
Effectors
actions

Agents in Artificial Intelligence

An AI system can be defined as the study of the rational agent and its environment. The agents sense the environment through sensors and act on their environment through actuators. An AI agent can have mental properties such as knowledge, belief, intention, etc.

What is an Agent?

An agent can be anything that perceive its environment through sensors and act upon that environment through actuators. An Agent runs in the cycle of **perceiving**, **thinking**, and **acting**. An agent can be:

- **Human-Agent:** A human agent has eyes, ears, and other organs which work for sensors and hand, legs, vocal tract work for actuators.
- **Robotic Agent:** A robotic agent can have cameras, infrared range finder, NLP for sensors and various motors for actuators.
- **Software Agent:** Software agent can have keystrokes, file contents as sensory input and act on those inputs and display output on the screen.

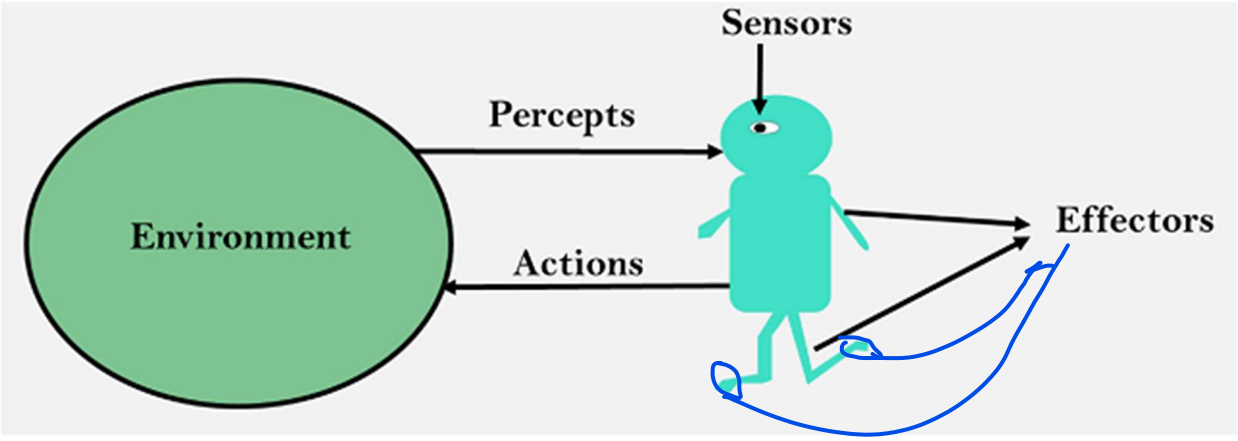
Hence the world around us is full of agents such as thermostat, cellphone, camera, and even we are also agents.

Before moving forward, we should first know about 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.



Intelligent Agents:

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. A thermostat is an example 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 an AI agent must be a rational action.

Rational Agent:

A rational agent is an agent which has clear preference, models uncertainty, and acts in a way to maximize its performance measure with all possible actions.

A rational agent is said to perform the right things. AI is about creating rational agents to use for game theory and decision theory for various real-world scenarios.

For an AI agent, the rational action is most important because in AI reinforcement learning algorithm, for each best possible action, agent gets the positive reward and for each wrong action, an agent gets a negative reward.

Rationality:

The rationality of an agent is measured by its performance measure. Rationality can be judged on the basis of following points:

- Performance measure which defines the success criterion.
- Agent prior knowledge of its environment.
- Best possible actions that an agent can perform.
- The sequence of percepts.

Structure of an AI Agent

The task of AI is to design an agent program which implements the agent function. The structure of an intelligent agent is a combination of architecture and agent program. It can be viewed as:

1. Agent = Architecture + Agent program

Following are the main three terms involved in the structure of an AI agent:

Architecture: Architecture is machinery that an AI agent executes on.

Agent Function: Agent function is used to map a percept to an action.

1. $f:P^* \rightarrow A$

Agent program: Agent program is an implementation of agent function. An agent program executes on the physical architecture to produce function f.

Types of AI Agents

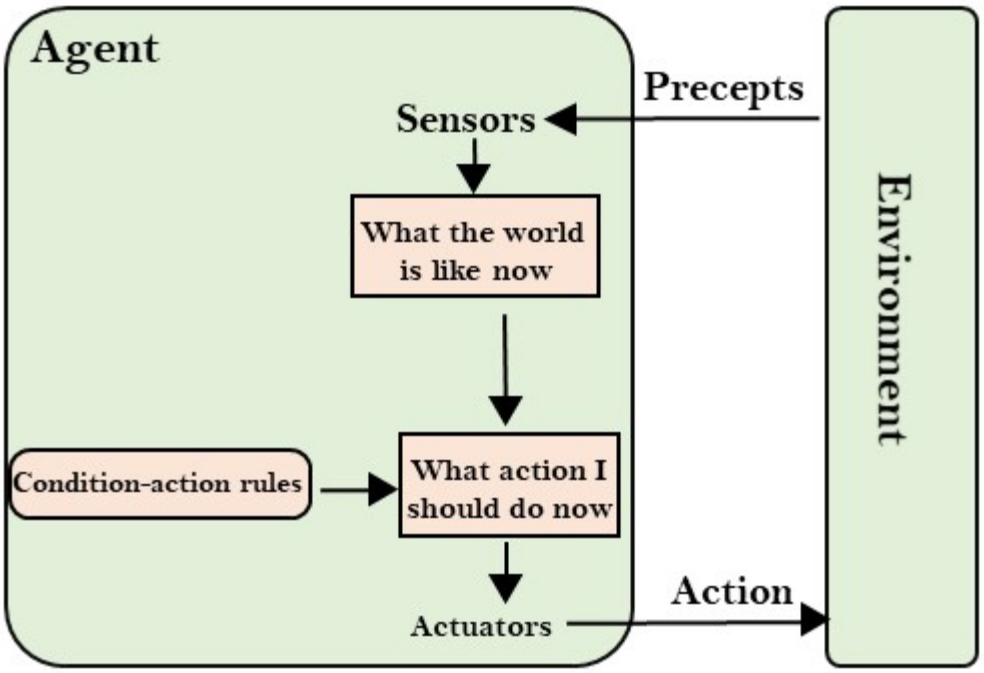
Agents can be grouped into five classes based on their degree of perceived intelligence and capability. All these agents can improve their performance and generate better action over the time. These are given below:

- Simple Reflex Agent

- Model-based reflex agent
- Goal-based agents
- Utility-based agent
- Learning agent

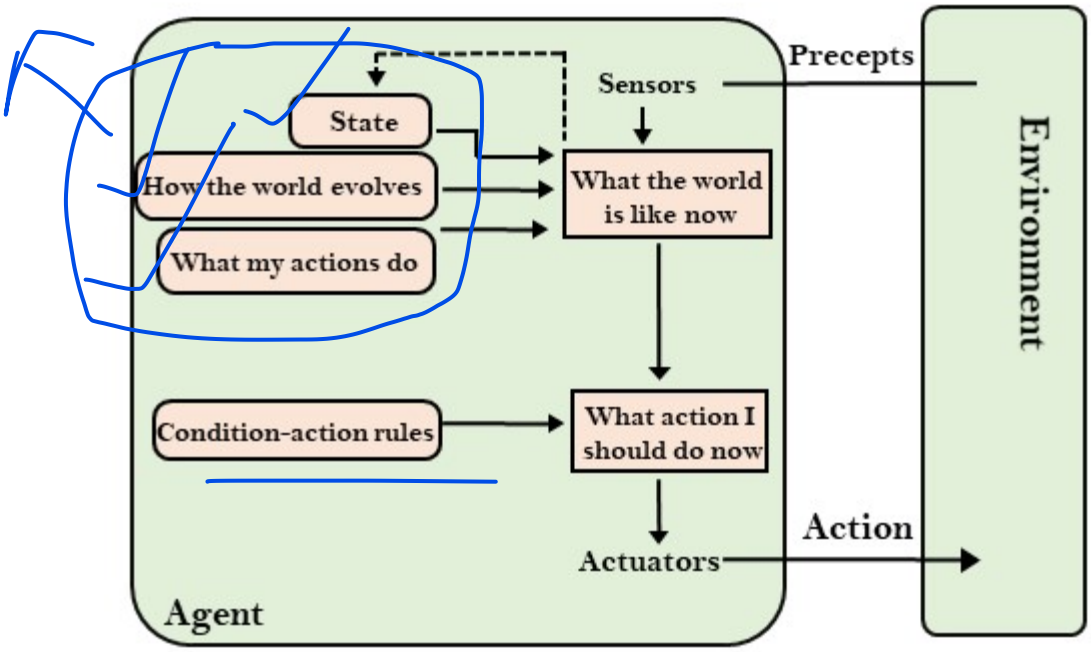
1. Simple Reflex agent:

- The Simple reflex agents are the simplest agents. These agents take decisions on the basis of the current percepts and ignore the rest of the percept history.
- These agents only succeed in the fully observable environment.
- The Simple reflex agent does not consider any part of percepts history during their decision and action process.
- The Simple reflex agent works on Condition-action rule, which means it maps the current state to action. Such as a Room Cleaner agent, it works only if there is dirt in the room.
- Problems for the simple reflex agent design approach:
    - They have very limited intelligence
    - They do not have knowledge of non-perceptual parts of the current state
    - Mostly too big to generate and to store.
    - Not adaptive to changes in the environment.
- **Condition-Action Rule** – It is a rule that maps a state (condition) to an action.



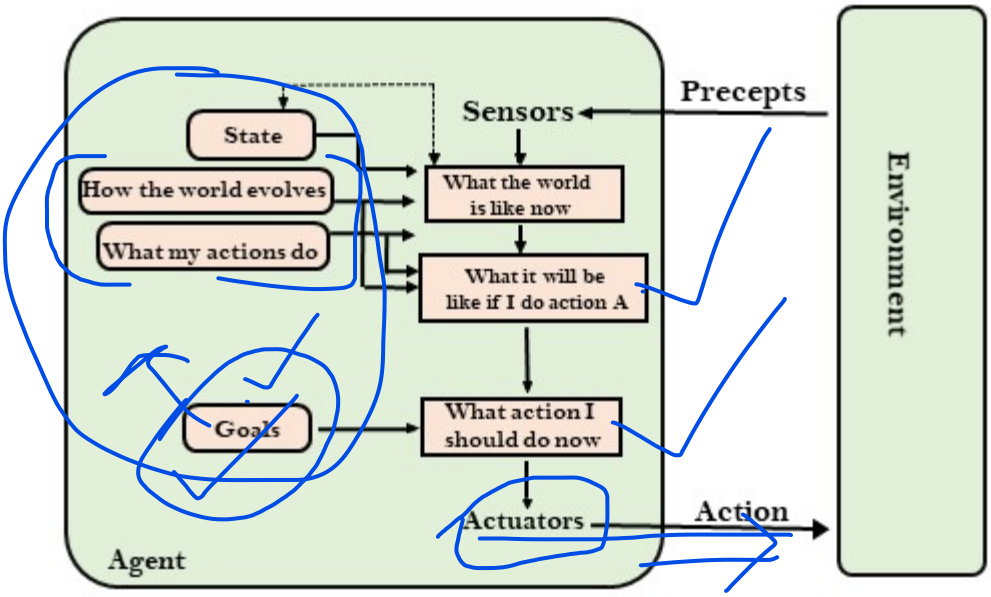
2. Model-based reflex agent

- The Model-based agent can work in a partially observable environment, and track the situation.
- A model-based agent has two important factors:
  - **Model:** It is knowledge about "how things happen in the world," so it is called a Model-based agent.
  - **Internal State:** It is a representation of the current state based on percept history.
- These agents have the model, "which is knowledge of the world" and based on the model they perform actions.
- Updating the agent state requires information about:
  - How the world evolves
  - How the agent's action affects the world.

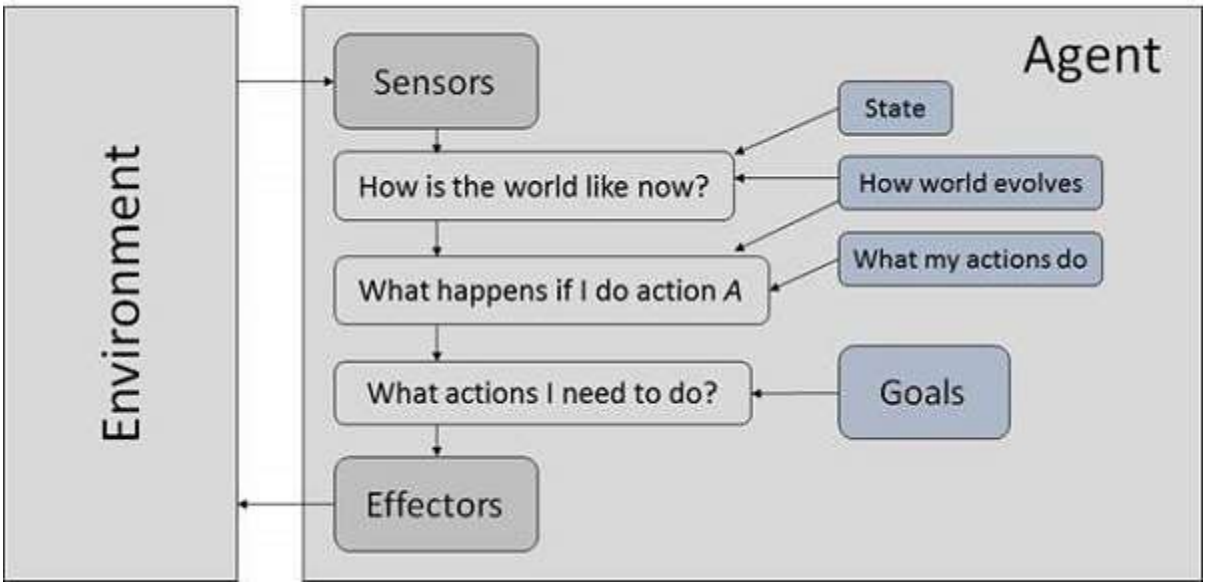


3. Goal-based agents

- The knowledge of the current state environment is not always sufficient to decide for an agent to what to do.
- The agent needs to know its goal which describes desirable situations.
- Goal-based agents expand the capabilities of the model-based agent by having the "goal" information.
- They choose an action, so that they can achieve the goal.
- These agents may have to consider a long sequence of possible actions before deciding whether the goal is achieved or not. Such considerations of different scenario are called searching and planning, which makes an agent proactive.

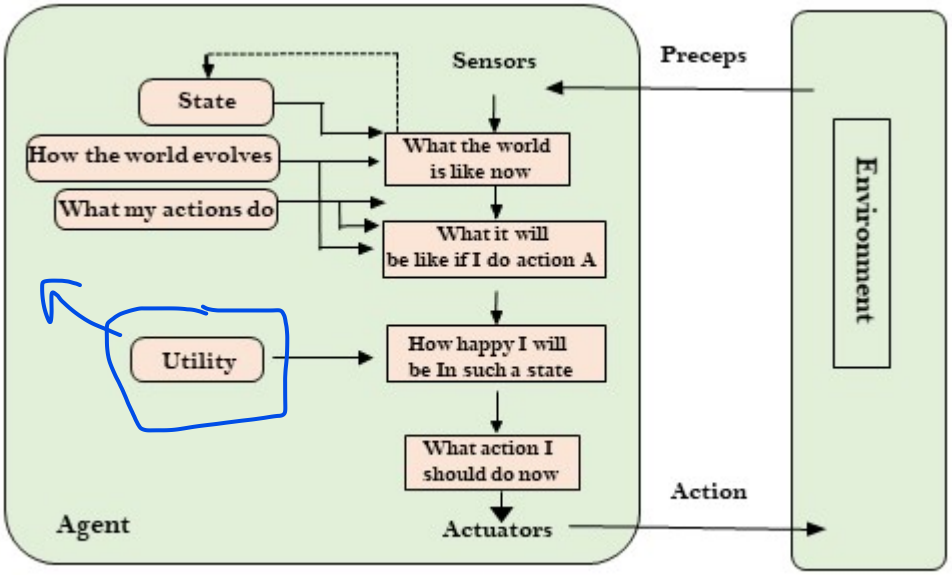


Goal – It is the description of desirable situations.



4. Utility-based agents

- These agents are similar to the goal-based agent but provide an extra component of utility measurement which makes them different by providing a measure of success at a given state.
- Utility-based agent act based not only goals but also the best way to achieve the goal.
- The Utility-based agent is useful when there are multiple possible alternatives, and an agent has to choose in order to perform the best action.
- The utility function maps each state to a real number to check how efficiently each action achieves the goals.

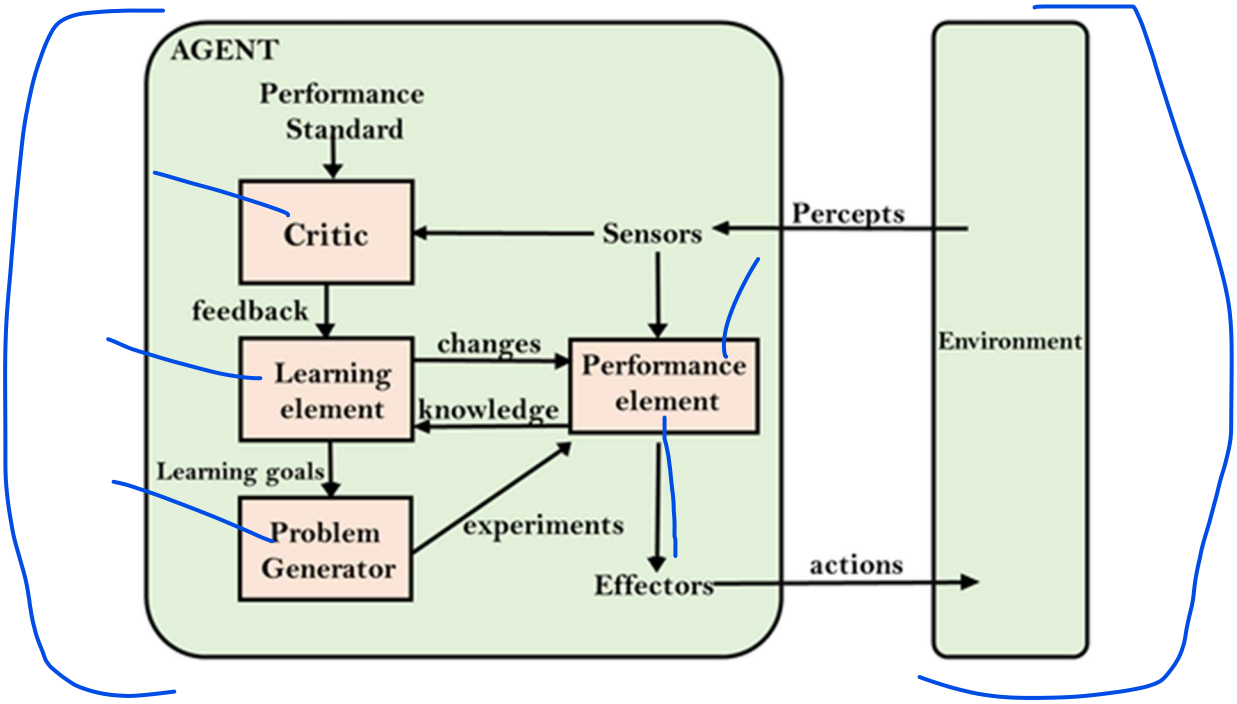


5. Learning Agents

- A learning agent in AI is the type of agent which can learn from its past experiences, or it has learning capabilities.
- It starts to act with basic knowledge and then able to act and adapt automatically through learning.
- A learning agent has mainly four conceptual components, which are:

1. **Learning element:** It is responsible for making improvements by learning from environment
2. **Critic:** Learning element takes feedback from critic which describes that how well the agent is doing with respect to a fixed performance standard.
3. **Performance element:** It is responsible for selecting external action
4. **Problem generator:** This component is responsible for suggesting actions that will lead to new and informative experiences.

- Hence, learning agents are able to learn, analyze performance, and look for new ways to improve the performance.



PEAS Representation

PEAS is a type of model on which an AI agent works upon. When we define an AI agent or rational agent, then we can group its properties under PEAS representation model. It is made up of four words:

- **P:** Performance measure
- **E:** Environment
- **A:** Actuators

- **S:** Sensors

Here performance measure is the objective for the success of an agent's behavior.

PEAS for self-driving cars:

Let's suppose a self-driving car then PEAS representation will be:

Performance: Safety, time, legal drive, comfort

Environment: Roads, other vehicles, road signs, pedestrian

Actuators: Steering, accelerator, brake, signal, horn

Sensors: Camera, GPS, speedometer, odometer, accelerometer, sonar.

Example of Agents with their PEAS representation

Agent	Performance measure	Environment	Actuators	Sensors
1. Medical Diagnose	○ Healthy patient ○ Minimized cost	○ Patient ○ Hospital ○ Staff	○ Tests ○ Treatments	Keyboard (Entry of symptoms)
2. Vacuum Cleaner	○ Cleanness ○ Efficiency ○ Battery life ○ Security	○ Room ○ Table ○ Wood floor ○ Carpet ○ Various obstacles	○ Wheels ○ Brushes ○ Vacuum Extractor	○ Camera ○ Dirt detection sen ○ Cliff sensor ○ Bump Sensor ○ Infrared Wall Sensor

Agent Environment in AI

An environment is everything in the world which surrounds the agent, but it is not a part of an agent itself. An environment can be described as a situation in which an agent is present.

The environment is where agent lives, operate and provide the agent with something to sense and act upon it..

Features of Environment

As per Russell and Norvig, an environment can have various features from the point of view of an agent:

1. Fully observable vs Partially Observable
2. Static vs Dynamic
3. Discrete vs Continuous
4. Deterministic vs Stochastic
5. Single-agent vs Multi-agent
6. Episodic vs sequential
7. Known vs Unknown
8. Accessible vs Inaccessible

1. Fully observable vs Partially Observable:

- If an agent sensor can sense or access the complete state of an environment at each point of time then it is **a fully observable** environment, else it is **partially observable**.
- A fully observable environment is easy as there is no need to maintain the internal state to keep track history of the world.
- An agent with no sensors in all environments then such an environment is called as **unobservable**.

2. Deterministic vs Stochastic:

- If an agent's current state and selected action can completely determine the next state of the environment, then such environment is called a deterministic environment.
- A stochastic environment is random in nature and cannot be determined completely by an agent.
- In a deterministic, fully observable environment, agent does not need to worry about uncertainty.

3. Episodic vs Sequential:

- In an episodic environment, there is a series of one-shot actions, and only the current percept is required for the action.
- However, in Sequential environment, an agent requires memory of past actions to determine the next best actions.

4. Single-agent vs Multi-agent

- If only one agent is involved in an environment, and operating by itself then such an environment is called single agent environment.
- However, if multiple agents are operating in an environment, then such an environment is called a multi-agent environment.
- The agent design problems in the multi-agent environment are different from single agent environment.

5. Static vs Dynamic:

- If the environment can change itself while an agent is deliberating then such environment is called a dynamic environment else it is called a static environment.
- Static environments are easy to deal because an agent does not need to continue looking at the world while deciding for an action.
- However for dynamic environment, agents need to keep looking at the world at each action.

- Taxi driving is an example of a dynamic environment whereas Crossword puzzles are an example of a static environment.

6. Discrete vs Continuous:

- If in an environment there are a finite number of percepts and actions that can be performed within it, then such an environment is called a discrete environment else it is called continuous environment.
- A chess gamecomes under discrete environment as there is a finite number of moves that can be performed.
- A self-driving car is an example of a continuous environment.

7. Known vs Unknown

- Known and unknown are not actually a feature of an environment, but it is an agent's state of knowledge to perform an action.
- In a known environment, the results for all actions are known to the agent. While in unknown environment, agent needs to learn how it works in order to perform an action.
- It is quite possible that a known environment to be partially observable and an Unknown environment to be fully observable.

8. Accessible vs Inaccessible

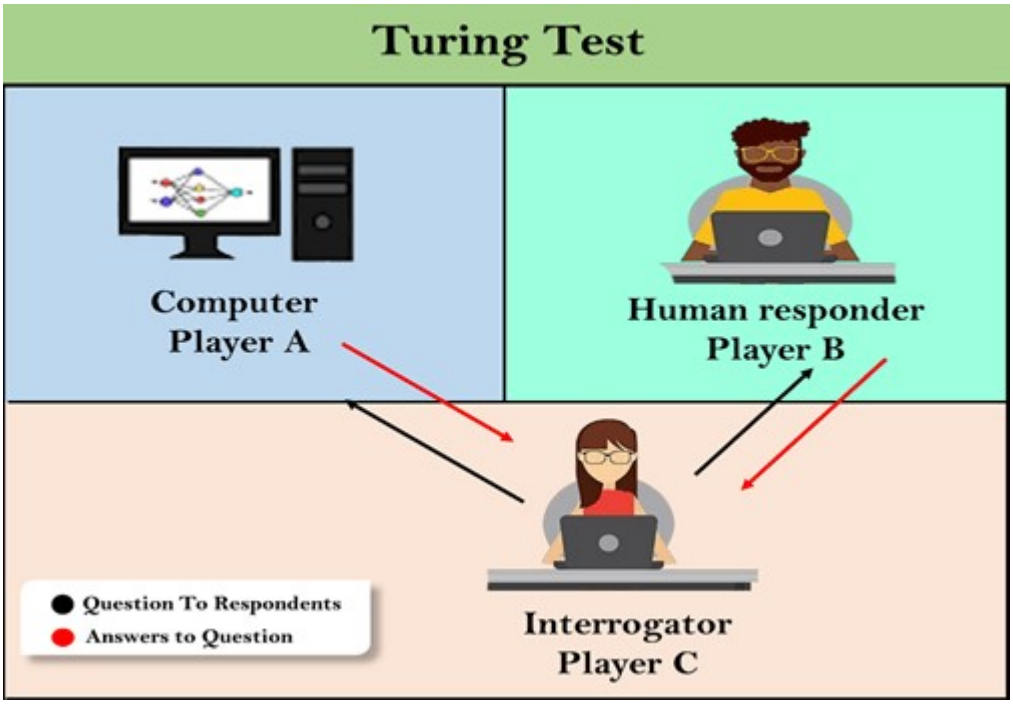
- If an agent can obtain complete and accurate information about the state's environment, then such an environment is called an Accessible environment else it is called inaccessible.
- An empty room whose state can be defined by its temperature is an example of an accessible environment.
- Information about an event on earth is an example of Inaccessible environment.

Turing Test in AI

- In 1950, Alan Turing introduced a test to check whether a machine can think like a human or not, this test is known as the Turing Test. In this test, Turing proposed

that the computer can be said to be an intelligent if it can mimic human response under specific conditions.

- Turing Test was introduced by Turing in his 1950 paper, "Computing Machinery and Intelligence," which considered the question, "Can Machine think?"



- The Turing test is based on a party game "Imitation game," with some modifications. This game involves three players in which one player is Computer, another player is human responder, and the third player is a human Interrogator, who is isolated from other two players and his job is to find that which player is machine among two of them.

- Consider, Player A is a computer, Player B is human, and Player C is an interrogator. Interrogator is aware that one of them is machine, but he needs to identify this on the basis of questions and their responses.

Chatbots to attempt the Turing test:

ELIZA: ELIZA was a Natural language processing computer program created by Joseph Weizenbaum. It was created to demonstrate the ability of communication between machine and humans. It was one of the first chatterbots, which has attempted the Turing Test.

Parry: Parry was a chatterbot created by Kenneth Colby in 1972. Parry was designed to simulate a person with **Paranoid schizophrenia**(most common chronic mental disorder). Parry was described as "ELIZA with attitude." Parry was tested using a variation of the Turing Test in the early 1970s.

Eugene Goostman: Eugene Goostman was a chatbot developed in Saint Petersburg in 2001. This bot has competed in the various number of Turing Test. In June 2012, at an event, Goostman won the competition promoted as largest-ever Turing test content, in which it has convinced 29% of judges that it was a human. Goostman resembled as a 13-year old virtual boy.

Features required for a machine to pass the Turing test:

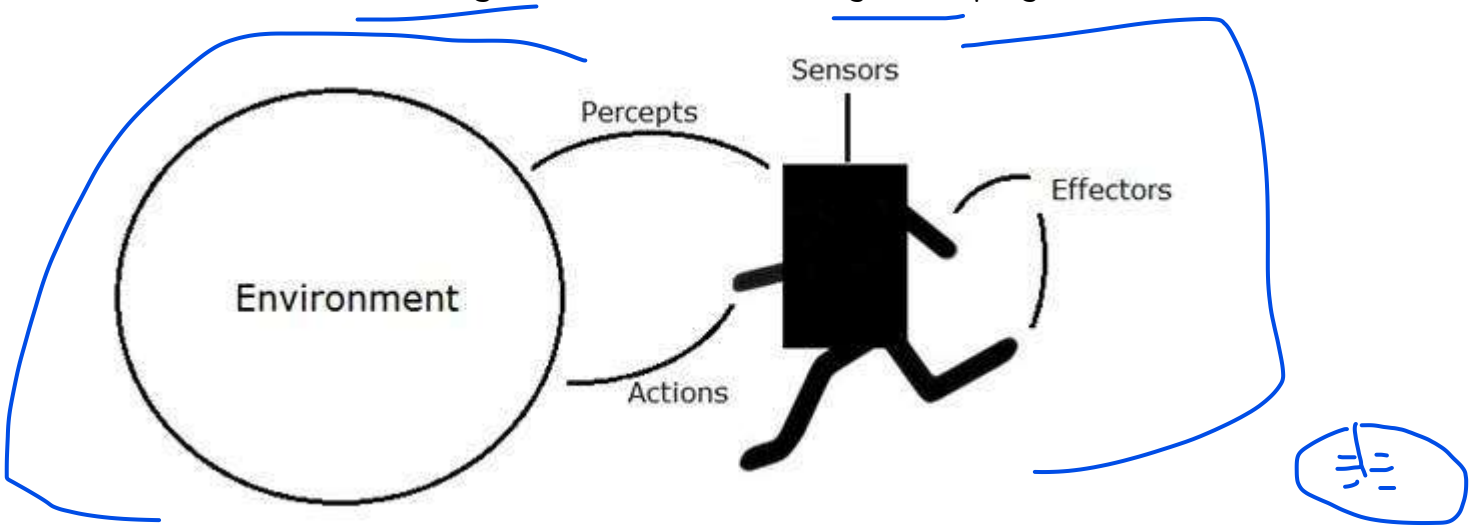
- **Natural language processing:** NLP is required to communicate with Interrogator in general human language like English.
- **Knowledge representation:** To store and retrieve information during the test.
- **Automated reasoning:** To use the previously stored information for answering the questions.
- **Machine learning:** To adapt new changes and can detect generalized patterns.
- **Vision (For total Turing test):** To recognize the interrogator actions and other objects during a test.
- **Motor Control (For total Turing test):** To act upon objects if requested.

An AI system is composed of an agent and its environment. The agents act in their environment. The environment may contain other agents.

What are Agent and Environment?

An **agent** is anything that can perceive its environment through **sensors** and acts upon that environment through **effectors.**

- A **human agent** has sensory organs such as eyes, ears, nose, tongue and skin parallel to the sensors, and other organs such as hands, legs, mouth, for effectors.
- A **robotic agent** replaces cameras and infrared range finders for the sensors, and various motors and actuators for effectors.
- A **software agent** has encoded bit strings as its programs and actions.



Agent Terminology

- **Performance Measure of Agent** – It is the criteria, which determines how successful an agent is.
- **Behavior of Agent** – It is the action that agent performs after any given sequence of percepts.
- **Percept** – It is agent's perceptual inputs at a given instance.
- **Percept Sequence** – It is the history of all that an agent has perceived till date.
- **Agent Function** – It is a map from the precept sequence to an action.

Rationality

Rationality is nothing but status of being reasonable, sensible, and having good sense of judgment.

Rationality is concerned with expected actions and results depending upon what the agent has perceived. Performing actions with the aim of obtaining useful information is an important part of rationality.

What is Ideal Rational Agent?

An ideal rational agent is the one, which is capable of doing expected actions to maximize its performance measure, on the basis of –

- Its percept sequence
- Its built-in knowledge base

Rationality of an agent depends on the following –

- The **performance measures**, which determine the degree of success.
- Agent's **Percept Sequence** till now.
- The agent's **prior knowledge about the environment**.
- The **actions** that the agent can carry out.

A rational agent always performs right action, where the right action means the action that causes the agent to be most successful in the given percept sequence. The problem the agent solves is characterized by **Performance Measure, Environment, Actuators, and Sensors (PEAS).**

The Structure of Intelligent Agents

Agent's structure can be viewed as –

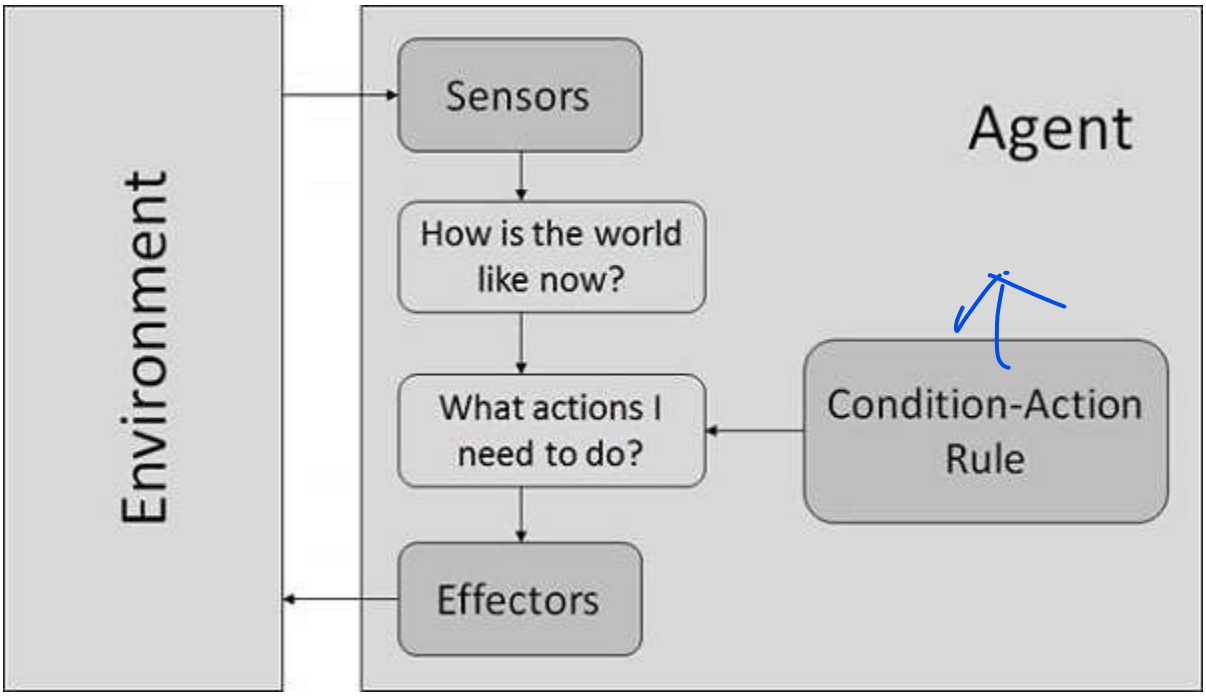
- Agent = Architecture + Agent Program
- Architecture = the machinery that an agent executes on.
- Agent Program = an implementation of an agent function.

Simple Reflex Agents

- They choose actions only based on the current percept.
- They are rational only if a correct decision is made only on the basis of current precept.

- Their environment is completely observable.

Condition-Action Rule – It is a rule that maps a state (condition) to an action.



Model Based Reflex Agents

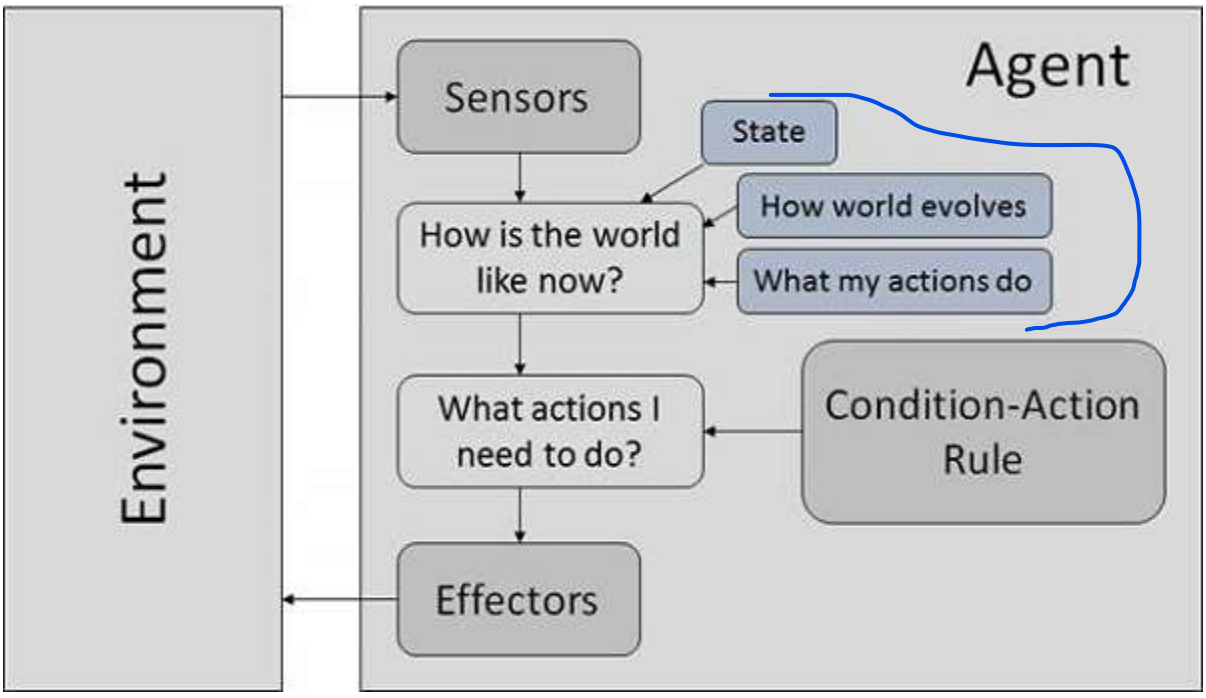
They use a model of the world to choose their actions. They maintain an internal state.

Model – knowledge about "how the things happen in the world".

Internal State – It is a representation of unobserved aspects of current state depending on percept history.

Updating the state requires the information about –

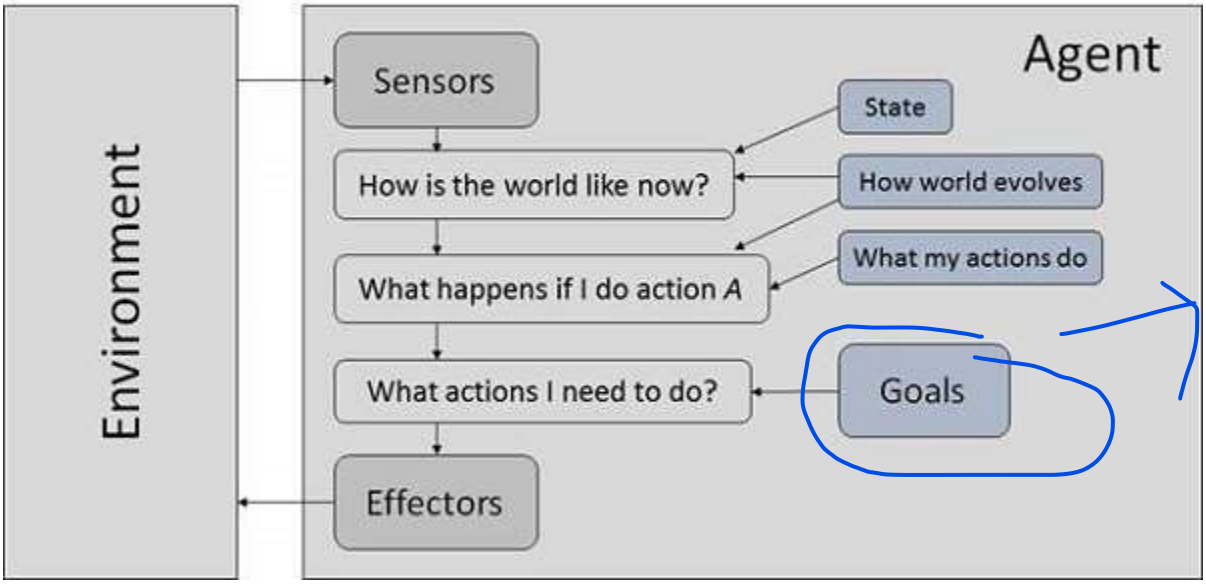
- How the world evolves.
- How the agent's actions affect the world.



Goal Based Agents

They choose their actions in order to achieve goals. Goal-based approach is more flexible than reflex agent since the knowledge supporting a decision is explicitly modeled, thereby allowing for modifications.

Goal – It is the description of desirable situations.

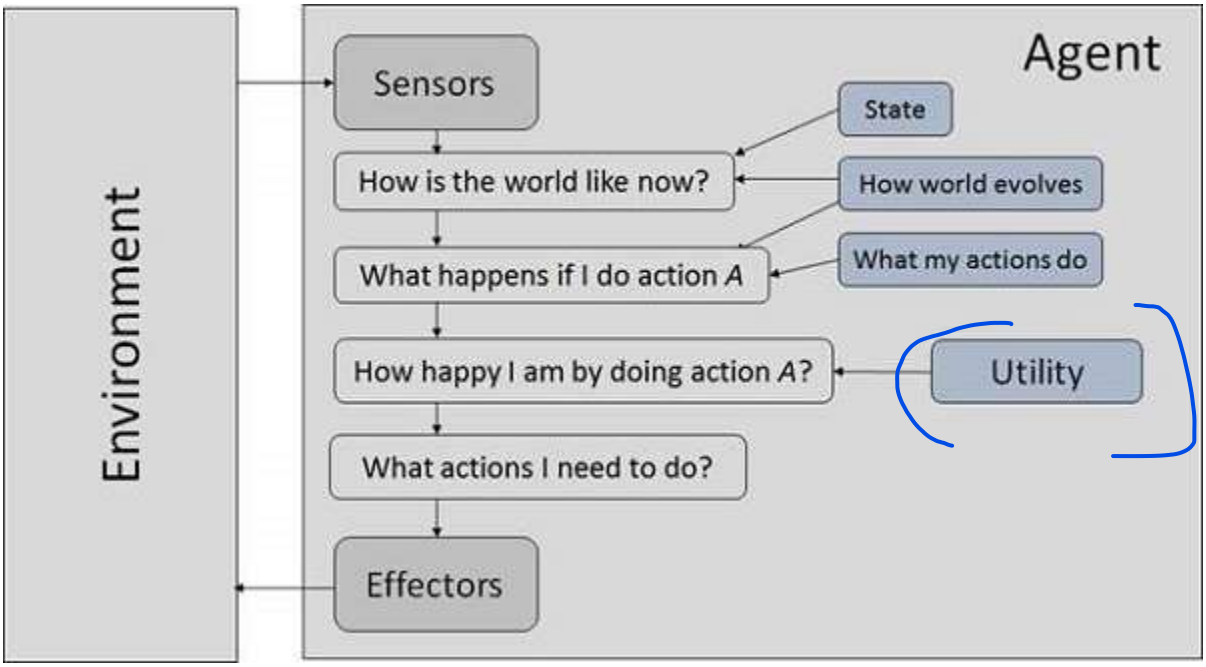


Utility Based Agents

They choose actions based on a preference (utility) for each state.

Goals are inadequate when –

- There are conflicting goals, out of which only few can be achieved.
- Goals have some uncertainty of being achieved and you need to weigh likelihood of success against the importance of a goal.



The Nature of Environments

Some programs operate in the entirely **artificial environment** confined to keyboard input, database, computer file systems and character output on a screen.

In contrast, some software agents (software robots or softbots) exist in rich, unlimited softbots domains. The simulator has a **very detailed, complex environment**. The software agent needs to choose from a long array of actions in real time. A softbot designed to scan the online preferences of the customer and show interesting items to the customer works in the **real** as well as an **artificial** environment.

The most famous **artificial environment** is the **Turing Test environment**, in which one real and other artificial agents are tested on equal ground. This is a very challenging environment as it is highly difficult for a software agent to perform as well as a human.

Turing Test

The success of an intelligent behavior of a system can be measured with Turing Test.

Two persons and a machine to be evaluated participate in the test. Out of the two persons, one plays the role of the tester. Each of them sits in different rooms. The tester is unaware of who is machine and who is a human. He interrogates the questions by typing and sending them to both intelligences, to which he receives typed responses.

This test aims at fooling the tester. If the tester fails to determine machine's response from the human response, then the machine is said to be intelligent.

Properties of Environment

The environment has multifold properties –

- **Discrete / Continuous** – If there are a limited number of distinct, clearly defined, states of the environment, the environment is discrete (For example, chess); otherwise it is continuous (For example, driving).
- **Observable / Partially Observable** – If it is possible to determine the complete state of the environment at each time point from the percepts it is observing; otherwise it is only partially observable.
- **Static / Dynamic** – If the environment does not change while an agent is acting, then it is static; otherwise it is dynamic.
- **Single agent / Multiple agents** – The environment may contain other agents which may be of the same or different kind as that of the agent.
- **Accessible / Inaccessible** – If the agent's sensory apparatus can have access to the complete state of the environment, then the environment is accessible to that agent.
- **Deterministic / Non-deterministic** – If the next state of the environment is completely determined by the current state and the actions of the agent, then the environment is deterministic; otherwise it is non-deterministic.
- **Episodic / Non-episodic** – In an episodic environment, each episode consists of the agent perceiving and then acting. The quality of its action depends just on the episode itself. Subsequent episodes do not depend on the actions in the previous episodes. Episodic environments are much simpler because the agent does not need to think ahead.

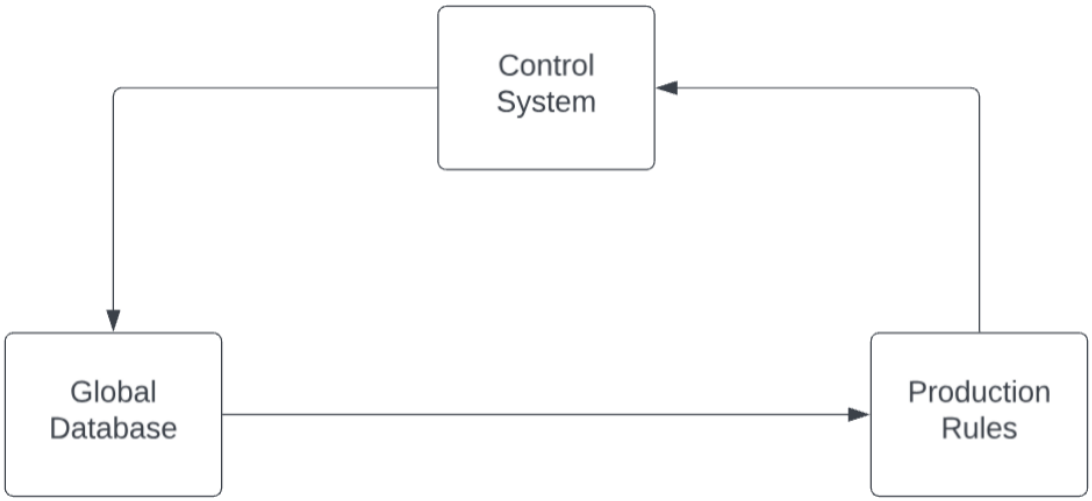
What is a Production System in AI?

A production system is based on a set of rules about behavior. These rules are a basic representation found helpful in expert systems, automated planning, and action selection.

Production system or production rule system is a computer program typically used to provide some form of artificial intelligence, which consists primarily of a set of rules about behavior but it also includes the mechanism necessary to follow those rules as the system responds to states of the world.

The major components of the Production System in Artificial Intelligence are:

- **Global Database:** The global database is the central data structure used by the production system in Artificial Intelligence.
- **Set of Production Rules:** The production rules operate on the global database. Each rule usually has a precondition that is either satisfied or not by the global database. If the precondition is satisfied, the rule is usually be applied. The application of the rule changes the database.
- **A Control System:** The control system then chooses which applicable rule should be applied and ceases computation when a termination condition on the database is satisfied. If multiple rules are to fire at the same time, the control system resolves the conflicts.



Production System Rules

Production System rules can be classified as:

You can represent the knowledge in a production system as a set of rules along with a control system and database. It can be written as:

- **Deductive Inference Rules**
- **Abductive Inference Rules**

If(Condition) Else (Condition)

Deductive Inference Rules	Abductive Inference Rules
Deductive inference rules are logic used in AI and knowledge-based systems. They facilitate deductive reasoning, which involves drawing specific conclusions from general premises or facts. In deductive reasoning, the conclusion is guaranteed to be true if the premises are true and the inference rule is valid. Modus Ponens and Modus Tollens are common deductive inference rules that help derive valid conclusions from given facts and rules.	Abductive inference rules are used in AI and reasoning systems to make educated guesses or hypotheses based on observed data or evidence. Abductive reasoning involves generating plausible explanations or hypotheses to explain the available information. Unlike deductive reasoning, abductive conclusions are not guaranteed true but are selected based on their likelihood, given the available evidence. Abductive inference is particularly useful in situations with incomplete or uncertain data, where the system needs to make the best possible guess or explanation.

The main features of the production system include:

1. Simplicity: The structure of each sentence in a production system is unique and uniform as they use the "IF-THEN" structure. This structure provides simplicity in knowledge representation. This feature of the production system improves the readability of production rules.

2. Modularity: This means the production rule code the knowledge available in discrete pieces. Information can be treated as a collection of independent facts which may be added or deleted from the system with essentially no deleterious side effects.

3. Modifiability: This means the facility for modifying rules. It allows the development of production rules in a skeletal form first and then it is accurate to suit a specific application.

4. Knowledge-intensive: The knowledge base of the production system stores pure knowledge. This part does not contain any type of control or programming information. Each production rule is normally written as an English sentence; the problem of semantics is solved by the very structure of the representation.

Classes of Production System in Artificial Intelligence

There are four major classes of Production System in Artificial Intelligence:

Monotonic Production System

In a monotonic production system, the laws and truths remain constant during execution. Once a fact is deduced, it and the corresponding rule stay fixed throughout the process. Consequently, this stability ensures predictability but may limit adaptability in dynamic environments where changes are frequent.

Partially Commutative Production System:

By contrast, in this type of system, the rules can be applied flexibly, allowing for some degree of adaptability while still maintaining certain constraints. This partial commutativity strikes a balance between stability and flexibility, making it useful in scenarios that require both some level of consistency and the ability to adjust to changes.

Non-Monotonic Production Systems

Conversely, non-monotonic production systems are more dynamic and adaptive. During execution, the system can add, modify, or retract rules. Therefore, this flexibility makes them excellent for situations where the knowledge base needs to change in response to shifting circumstances, offering high adaptability and responsiveness.

Commutative Systems

Finally, commutative systems have rules that can be applied in any sequence without changing the result. In circumstances where the sequence of rule application is not essential, this high degree of flexibility may be beneficial. Thus, commutative systems are ideal when the order of operations does not affect the outcome, providing significant operational flexibility.

Real-World Examples of AI Production Systems in Use

- **Customer Support Chatbots:** AI-powered chatbots in customer support systems use production rules to handle customer inquiries, provide answers, and escalate complex issues to human agents.
- **Fraud Detection Systems:** In financial institutions, AI production systems detect fraudulent activities by analyzing transaction data and applying predefined fraud detection rules.
- **Medical Diagnosis:** AI production systems are used in healthcare for medical diagnosis. They analyze patient symptoms, medical history, and test results to suggest possible diagnoses and treatment options.
- **Traffic Management:** Smart traffic management systems use AI production systems to optimize traffic flow by adjusting signal timings based on real-time traffic conditions and predefined rules.

Production System in Artificial Intelligence: Example

Problem Statement:

We have two jugs of capacity 5l and 3l (liter), and a tap with an endless supply of water. The objective is to obtain 4 liters exactly in the 5-liter jug with the minimum steps possible.

- Fill the 5-liter jug from the tap
- Empty the 5-liter jug
- Fill the 3-liter jug from the tap
- Empty the 3-liter jug
- Then, empty the 3-liter jug to 5 liter
- Empty the 5-liter jug to 3 liter

- Pour water from 3 liters to 5 liter
- Pour water from 5 liters to 3 liters but do not empty

An Introduction to Problem-Solving using Search Algorithms

Search algorithms are one of the most important areas of Artificial Intelligence. This topic will explain all about the search algorithms in AI.

Problem-solving agents:

In Artificial Intelligence, Search techniques are universal problem-solving methods. **Rational agents** or **Problem-solving agents** in AI mostly used these search strategies or algorithms to solve a specific problem and provide the best result. Problem-solving agents are the goal-based agents and use atomic representation. In this topic, we will learn various problem-solving search algorithms.

Artificial Intelligence is the study of building agents that act rationally. Most of the time, these agents perform some kind of search algorithm in the background in order to achieve their tasks.
A search problem consists of:

- **A State Space.** Set of all possible states where you can be.
- **A Start State.** The state from where the search begins.
- **A Goal State.** A function that looks at the current state returns whether or not it is the goal state.

The process of problem-solving using searching consists of the following steps.

- Define the problem
- Analyze the problem
- Identification of possible solutions
- Choosing the optimal solution
- Implementation

There are basically three types of problem in artificial intelligence:

1. **Ignorable:** In which solution steps can be ignored.
2. **Recoverable:** In which solution steps can be undone.
3. **Irrecoverable:** Solution steps cannot be undo.

Steps problem-solving in AI:

The problem of AI is directly associated with the nature of humans and their activities. So we need a number of finite steps to solve a problem which makes human easy works.
These are the following steps which require to solve a problem :

- **Problem definition:** Detailed specification of inputs and acceptable system solutions.
- **Problem analysis:** Analyse the problem thoroughly.
- **Knowledge Representation:** collect detailed information about the problem and define all possible techniques.
- **Problem-solving:** Selection of best techniques.

Components to formulate the associated problem:

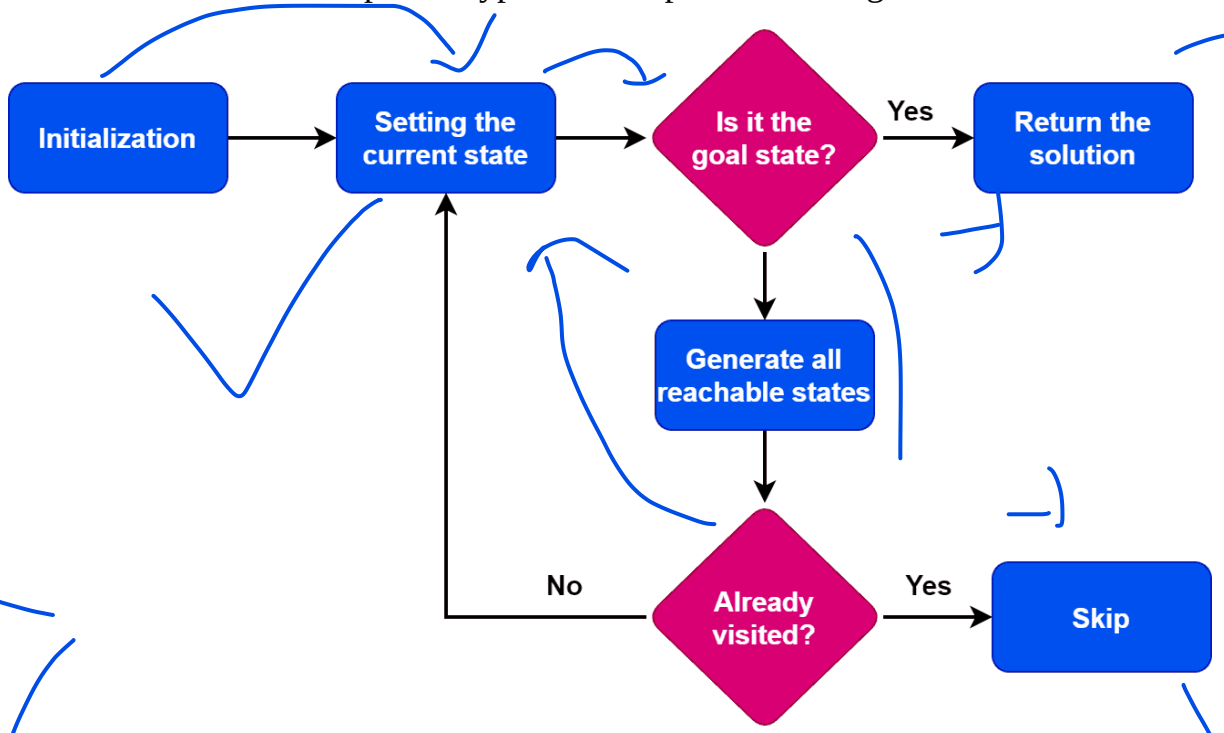
- **Initial State:** This state requires an initial state for the problem which starts the AI agent towards a specified goal. In this state new methods also initialize problem domain solving by a specific class.
- **Action:** This stage of problem formulation works with function with a specific class taken from the initial state and all possible actions done in this stage.
- **Transition:** This stage of problem formulation integrates the actual action done by the previous action stage and collects the final stage to forward it to their next stage.
- **Goal test:** This stage determines that the specified goal achieved by the integrated transition model or not, whenever the goal achieves stop the action and forward into the next stage to determines the cost to achieve the goal.
- **Path costing:** This component of problem-solving numerical assigned what will be the cost to achieve the goal. It requires all hardware software and human working cost.

State Space Search in Artificial Intelligence

- A state space is a mathematical representation of a problem that defines all possible states that the problem can be in. Furthermore, in search algorithms, we use a state space to represent the current state of the problem, the initial state, and the goal state.
- The state space size can greatly affect a search algorithm's efficiency. Hence, it's important to choose an appropriate representation and search strategy to efficiently search the state space.
- State space search is a strategy used in AI to find a path to a goal when there are many possibilities to consider. It's like having many doors to open, with each door leading to a room full of other doors. Some paths lead to dead ends, some go in circles, and some lead to the goal. The "state space" is the name for all the rooms and doors combined.
- AI uses this method to do things like planning the best route for delivery trucks, figuring out the moves in a game, or even making decisions about investments. It's a fundamental part of how AI systems think and make decisions.

Steps

Now let's discuss the steps of a typical state space search algorithm:



- To begin the search process, we set the current state to the initial state.
- We then check if the current state is the goal state. If it is, we terminate the algorithm and return the result.
- If the current state is not the goal state, we generate the set of possible successor states that can be reached from the current state.
- For each successor state, we check if it has already been visited. If it has, we skip it, else we add it to the queue of states to be visited.
- Next, we set the next state in the queue as the current state and check if it's the goal state. If it is, we return the result. If not, we repeat the previous step until we find the goal state or explore all the states.
- If all possible states have been explored and the goal state still needs to be found, we return with no solution.

Applications of State Space Search

State space search is a fundamental concept in artificial intelligence that has a wide range of applications. Here are some of the key areas where state space search is utilized:

1. Puzzle Solving:

State space search algorithms are often used to solve complex puzzles like the Rubik's Cube, sliding tile puzzles, and the water jug problem we discussed earlier. These puzzles can be represented as a state space, and algorithms can be applied to find solutions.

2. Pathfinding:

In video games and robotics, state space search is used for pathfinding – determining the shortest or most cost-effective path from one point to another. Algorithms like A* and Dijkstra's are used in maps and game levels to find paths considering various terrains and obstacles.

3. Planning and Scheduling:

AI planning involves creating a sequence of actions to achieve a specific goal. State space search helps in finding the best sequence of events in logistics, production schedules, or even in AI for playing strategy games.

4. Natural Language Processing:

State space search can be used in parsing algorithms for natural language processing, where the states represent possible interpretations of the sentences, and the search is for the most likely meaning.

5. Machine Learning:

In machine learning, especially in reinforcement learning, state space search helps in exploring different states and actions to maximize the reward function.

6. Problem Solving in AI:

State space search is a general problem-solving approach in AI. It is used in expert systems, theorem proving, and even in medical diagnosis systems where different states represent the presence or absence of symptoms and diseases.

7. Robotics:

Robots use state space search to navigate and interact with their environment. They need to understand the current state, explore possible next states, and choose the best action to perform tasks.

8. Automated Reasoning:

State space search is used in automated reasoning, where a system needs to infer new knowledge from the known facts. This is crucial in fields like law and computer-aided verification.

9. Optimization Problems:

Many optimization problems can be framed as state space searches, where the goal is to find the state that maximizes or minimizes a particular function.

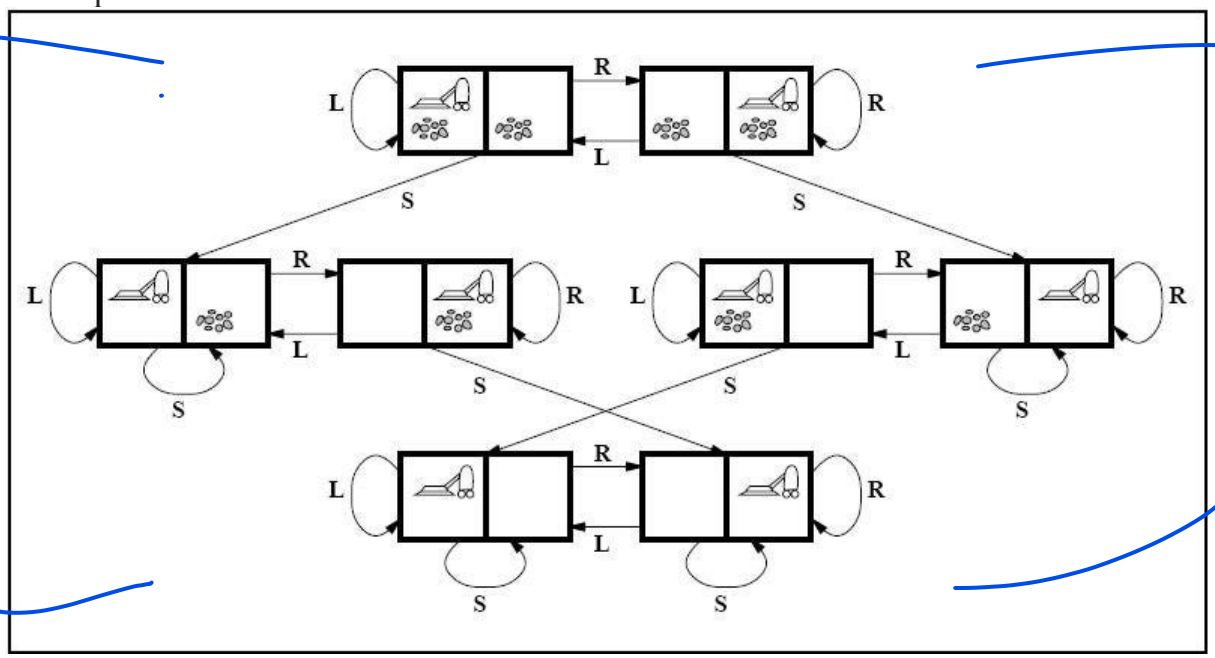
10. Quantum Computing:

In quantum computing, state space search can be used to explore the possibilities of quantum states to solve problems that are intractable on classical computers.

These applications demonstrate the versatility of state space search in solving a variety of problems by systematically exploring possible states and actions to find an optimal solution or to prove that no solution exists.

An Example Problem Formulation

Let us take the example of vacuum world that was introduced in the starting of this series, There is a vacuum cleaner agent and it can move left or right and its jump is to suck up the dirt from the floor.



State space for vacuum world.

The problem for vacuum world can be formulated as follows:

States: The state is determined by both the agent location and the dirt location. The agent is in one of two locations, each of which might or might not contain dirt. Therefore, there are 2 x 2^2 = **8** possible world states.

A larger environment would have n x 2 to the power of n states.

Initial State: Any state can be assigned as the initial state in this case.

Action: In this environment there are three actions, *Move Left , Move Right , Suck up the dirt.*

Transition Model: All the actions have expected effects, except for when the agent is in leftmost square and the action is *Left*, when the agent is in rightmost square and the action is *Right* and the square is clean when the action is to *Suck.*

Goal Test: Goal test checks whether all the squares are clean.

Path Cost: Each step costs 1, so the path cost is the number of steps in the path.

Properties of search algorithms

Completeness

A search algorithm is said to be complete when it gives a solution or returns any solution for a given random input.

Optimality
If a solution found is best (lowest path cost) among all the solutions identified, then that solution is said to be an optimal one.

Time complexity
The time taken by an algorithm to complete its task is called time complexity. If the algorithm completes a task in a lesser amount of time, then it is an efficient one.

Space complexity
It is the maximum storage or memory taken by the algorithm at any time while searching.

Types of search algorithms

- Uninformed search
- Informed search

Uninformed search algorithms

The uninformed search algorithm does not have any domain knowledge such as closeness, location of the goal state, etc. it behaves in a brute-force way.
It only knows the information about how to traverse the given tree and how to find the goal state. This algorithm is also known as the Blind search algorithm or Brute -Force algorithm.

The uninformed search strategies are of six types.

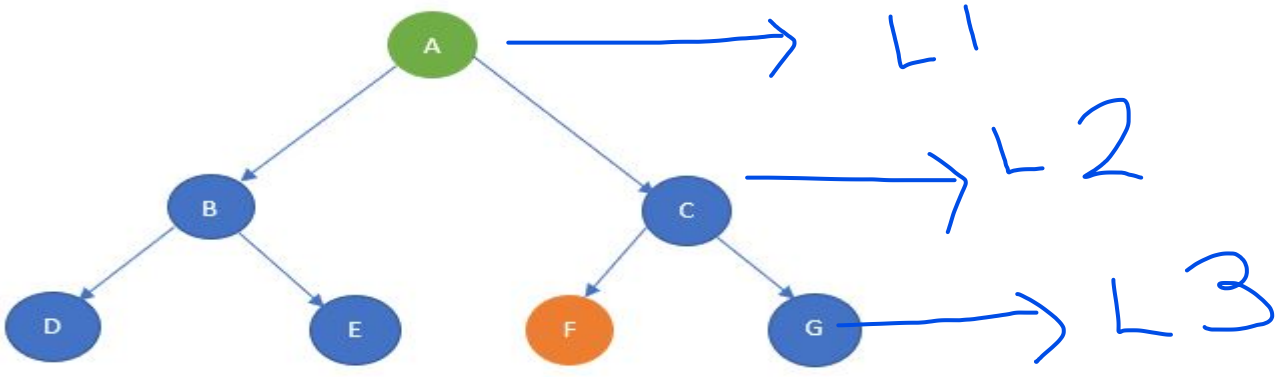
- Breadth-first search
- Depth-first search
- Depth-limited search
- Iterative deepening depth-first search
- Bidirectional search
- Uniform cost search

1. Breadth-first search

It is of the most common search strategies. It generally starts from the root node and examines the neighbor nodes and then moves to the next level. It uses First-in First-out (FIFO) strategy as it gives the shortest path to achieving the solution.

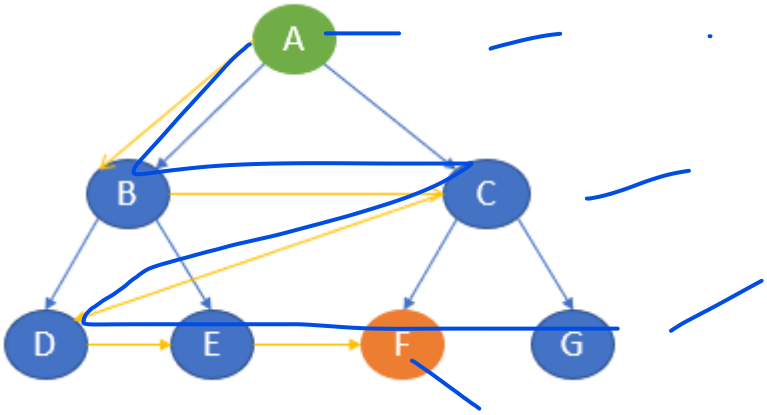
BFS is used where the given problem is very small and space complexity is not considered.

Now, consider the following tree.



The BFS algorithm starts with the start state and then goes to the next level and visits the node until it reaches the goal state.

In this example, it starts from A and then travel to the next level and visits B and C and then travel to the next level and visits D, E, F and G. Here, the goal state is defined as F. So, the traversal will stop at F.



The path of traversal is: A —-> B —-> C —-> D —-> E —-> F

Advantages of BFS

- BFS will never be trapped in any unwanted nodes.
- If the graph has more than one solution, then BFS will return the optimal solution which provides the shortest path.

Disadvantages of BFS

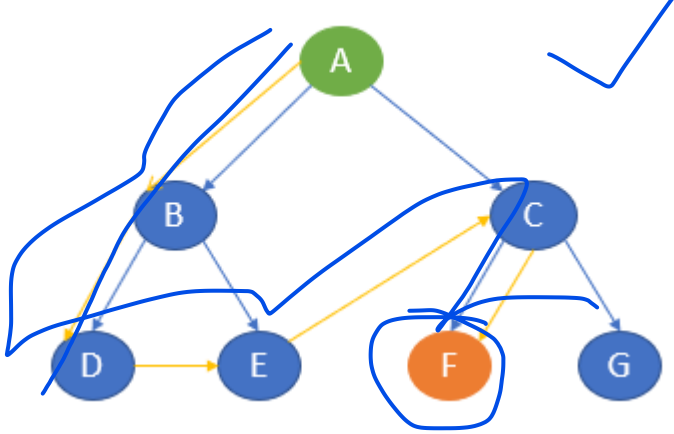
- BFS stores all the nodes in the current level and then go to the next level. It requires a lot of memory to store the nodes.
- BFS takes more time
  to reach the goal state which is far away.

2. Depth-first search

The depth-first search uses Last-in, First-out (LIFO) strategy and hence it can be implemented by using stack. DFS uses backtracking. That is, it starts from the initial state and explores each path to its greatest depth before it moves to the next path.

DFS will follow: Root node —-> Left node —-> Right node

Here, it starts from the start state A and then travels to B and then it goes to D. After reaching D, it backtracks to B. B is already visited, hence it goes to the next depth E and then backtracks to B. as it is already visited, it goes back to A. A is already visited. So, it goes to C and then to F. F is our goal state and it stops there.



The path of traversal is: A —-> B —-> D —-> E —-> C —-> F

Advantages of DFS

- It takes lesser memory as compared to BFS.
- The time complexity is lesser when compared to BFS.
- DFS does not require much more search.

Disadvantages of DFS

- DFS does not always guarantee to give a solution.
- As DFS goes deep down, it may get trapped in an infinite loop.

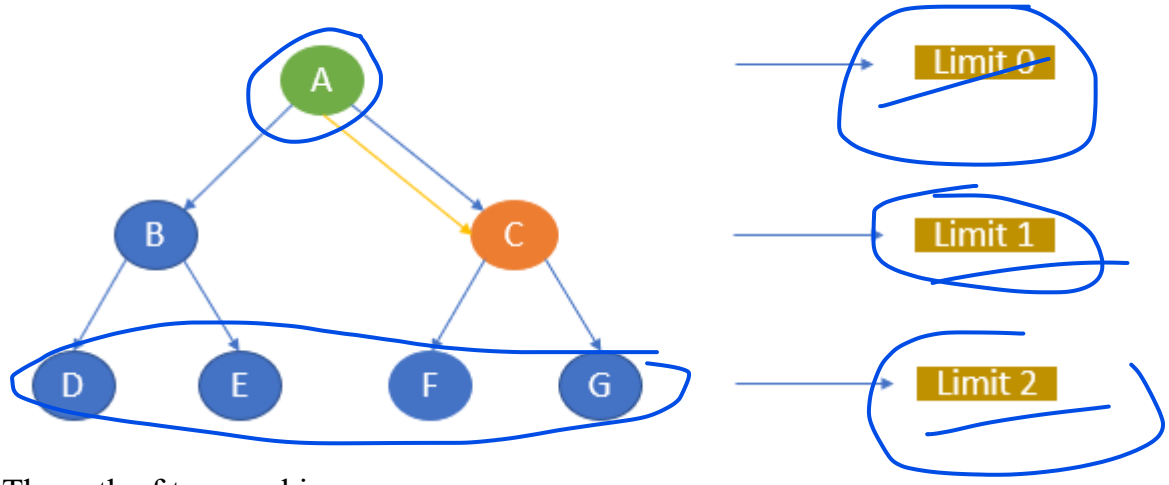
3. Depth-limited search

Depth-limited works similarly to depth-first search. The difference here is that depth-limited search has a pre-defined limit up to which it can traverse the nodes. Depth-limited search solves one of the drawbacks of DFS as it does not go to an infinite path.

Now, consider the same example.

Let's take A as the start node and C as the goal state and limit as 1.

The traversal first starts with node A and then goes to the next level 1 and the goal state C is there. It stops the traversal.

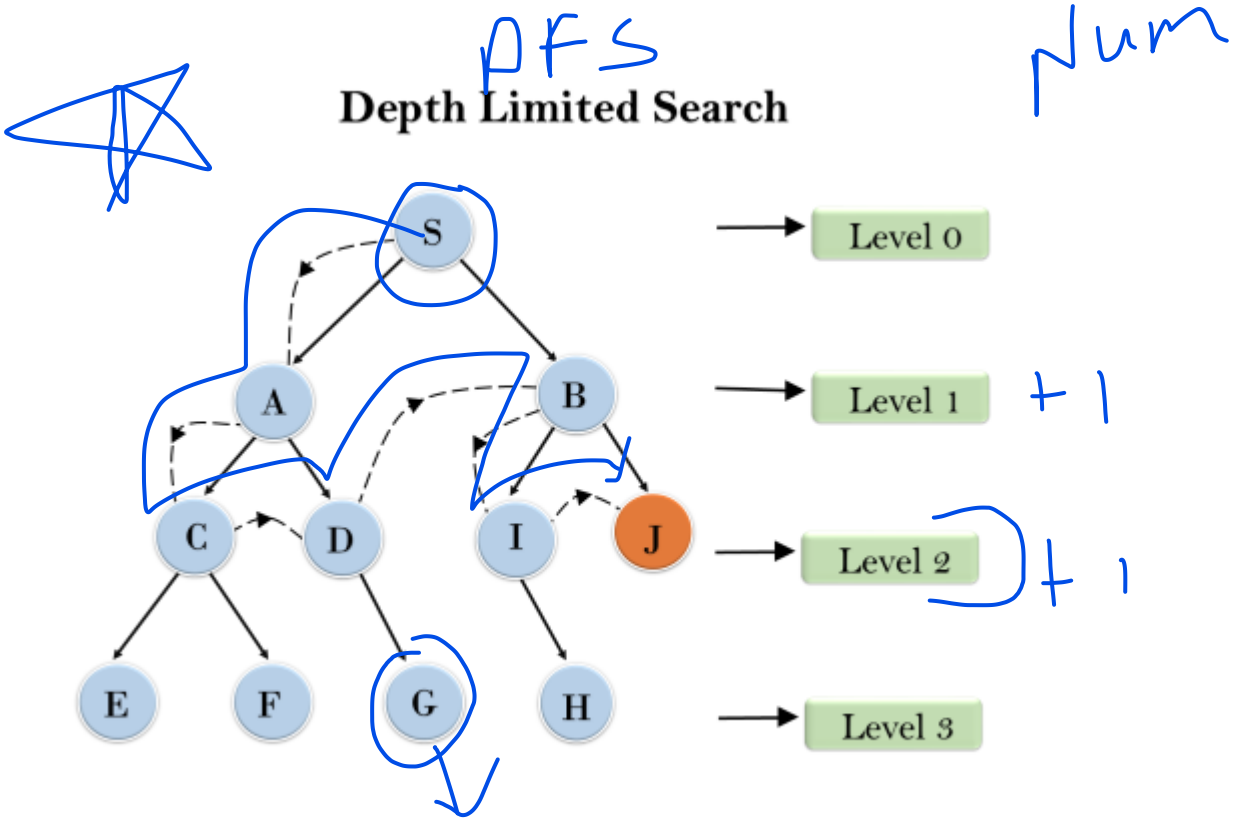


The path of traversal is:

A —-> C

If we give C as the goal node and the limit as 0, the algorithm will not return any path as the goal node is not available within the given limit.

If we give the goal node as F and limit as 2, the path will be A, C, F.



Advantages of DLS

- It takes lesser memory when compared to other search techniques.

Disadvantages of DLS

- DLS may not offer an optimal solution if the problem has more than one solution.
- DLS also encounters incompleteness.

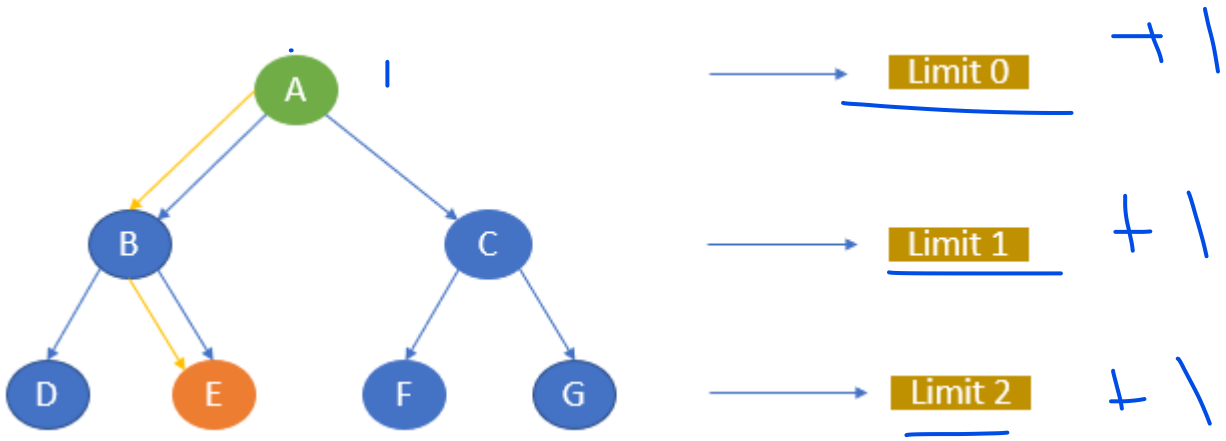
4. Iterative deepening depth-first search

Iterative deepening depth-first search is a combination of depth-first search and breadth-first search. IDDFS find the best depth limit by gradually adding the limit until the defined goal state is reached.

Let me try to explain this with the same example tree.

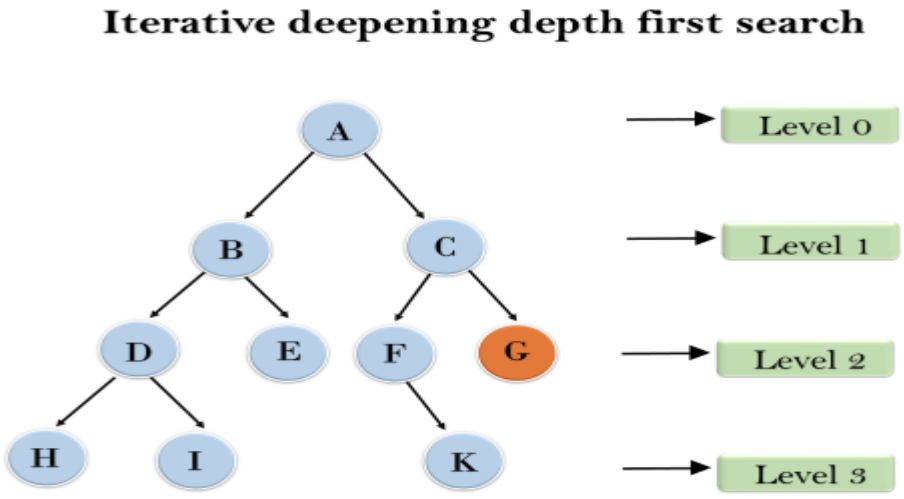
Consider, A as the start node and E as the goal node. Let the maximum depth be 2.

The algorithm starts
with A and goes to the next level and searches for E. If not found, it goes to
the next level and finds E.



The path of traversal is

A —-> B —-> E



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.

Advantages of IDDFS

- IDDFS has the advantages of both BFS and DFS.
- It offers fast search and uses memory efficiently.

Disadvantages of IDDFS

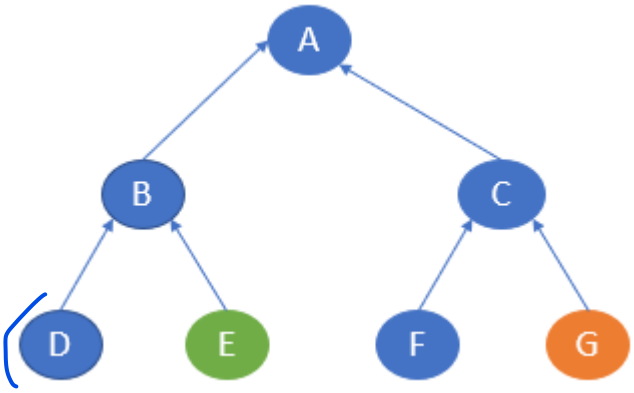
- It does all the works of the previous stage again and again.

5. Bidirectional search

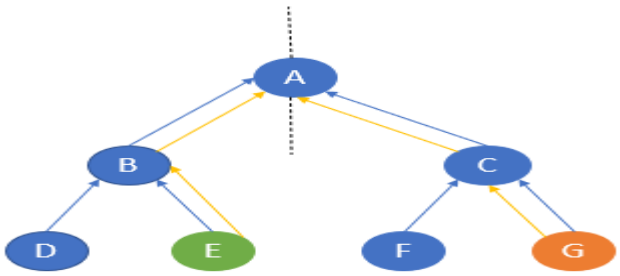
The bidirectional search algorithm is completely different from all other search strategies. It executes two simultaneous searches called forward-search and backwards-search and reaches the goal state. Here, the graph is divided into two smaller sub-graphs. In one graph, the search is started from the initial start state and in the other graph, the search is started from the goal state. When these two nodes intersect each other, the search will be terminated.

Bidirectional search requires both start and goal start to be well defined and the branching factor to be the same in the two directions.

Consider the below graph.



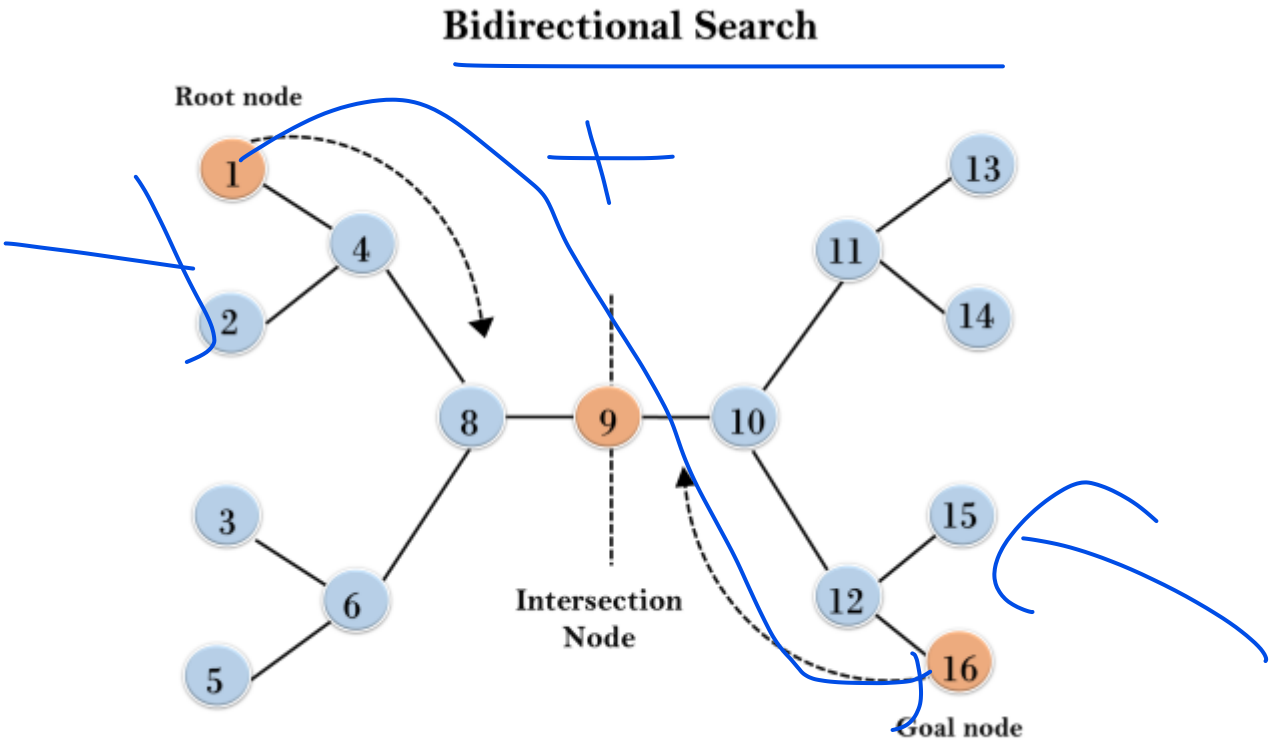
Here, the start state is E and the goal state is G. In one sub-graph, the search starts from E and in the other, the search starts from G. E will go to B and then A. G will go to C and then A. Here, both the traversal meets at A and hence the traversal ends.



The path of traversal is
E —-> B —-> A —-> C —-> G

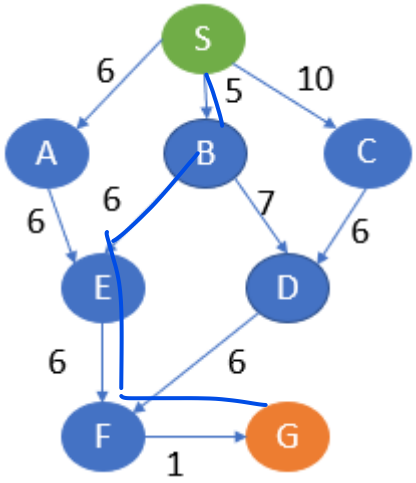
Example:

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.



6. Uniform cost search

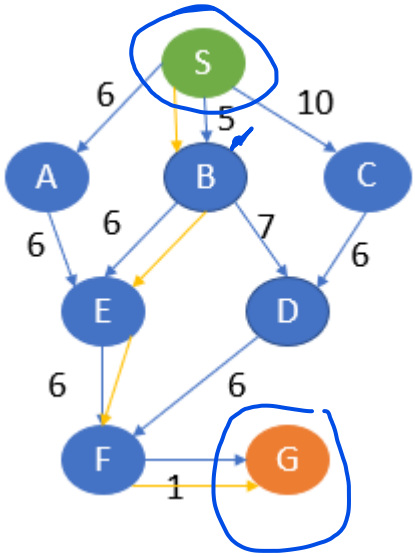
Uniform cost search is considered the best search algorithm for a weighted graph or graph with costs. It searches the graph by giving maximum priority to the lowest cumulative cost. Uniform cost search can be implemented using a priority queue.
Consider the below
graph where each node has a pre-defined cost.



Here, S is the start node and G is the goal node.

From S, G can be reached in the following ways.
S, A, E, F, G -> 19
S, B, E, F, G -> 18
S, B, D, F, G -> 19
S, C, D, F, G -> 23

Here, the path with the least cost is S, B, E, F, G.



Uninformed Search Algorithms

Uninformed search is a class of general-purpose search algorithms which operates in brute force-way. Uninformed search algorithms do not have additional information about state or search space other than how to traverse the tree, so it is also called blind search.

1. **Breadth-first Search**
2. **Depth-first Search**
3. **Depth-limited Search**
4. **Iterative deepening depth-first search**
5. Uniform Cost Search
6. **Means End Analysis**

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:

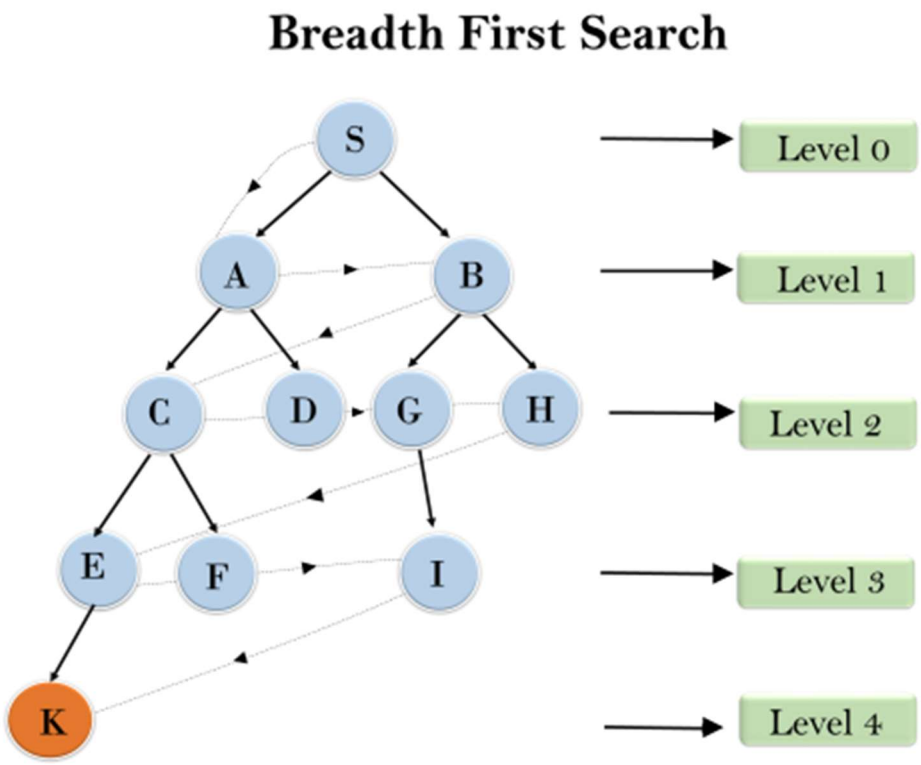
- BFS will provide a solution if any solution exists.
- If there are more than one solutions for a given problem, then BFS will provide the minimal solution which requires the least number of steps.

- **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:

1. 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.

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

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 isa 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.

Backtracking is an algorithm technique for finding all possible solutions using recursion.

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:

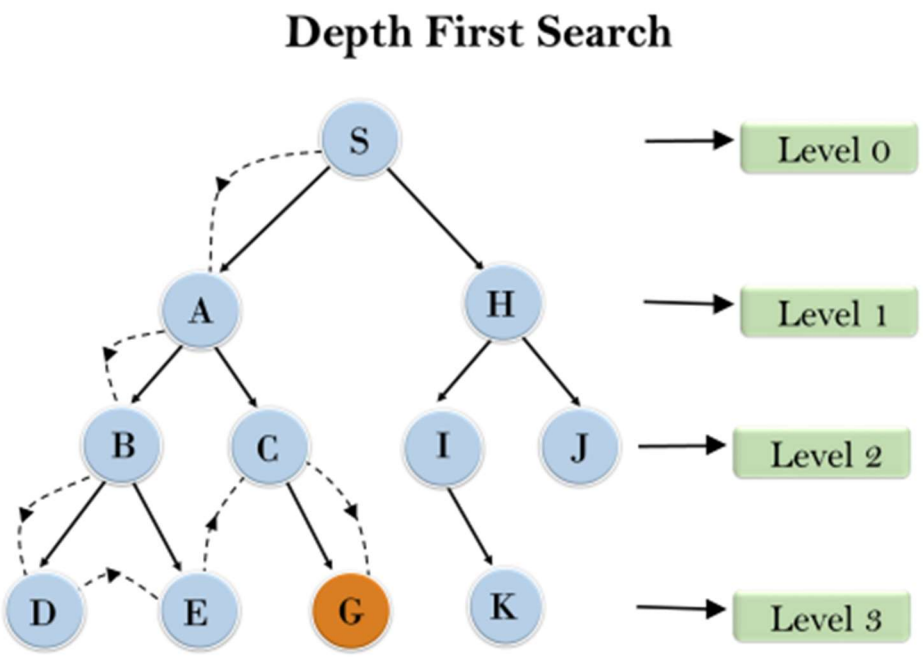
- here is the possibility that many states keep re-occurring, and there is no guarantee of finding the solution.
- DFS algorithm goes for deep down searching and sometime it may go to the infinite loop.

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:

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

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.

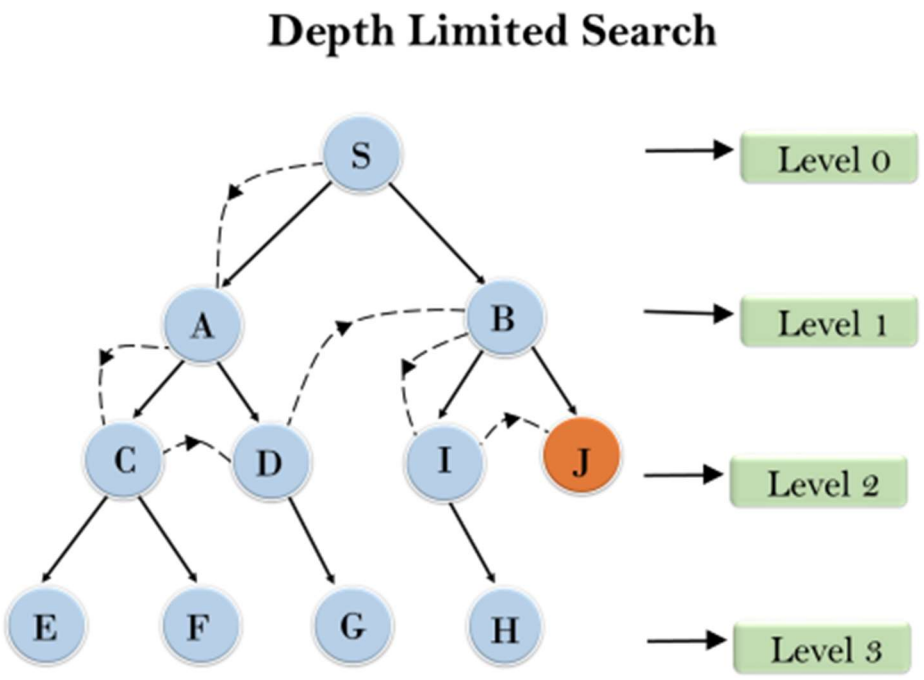
Advantages:

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.

Example:



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

Time Complexity: Time complexity of DLS algorithm is **$O(b^{\ell})$**.

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

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

4. 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:

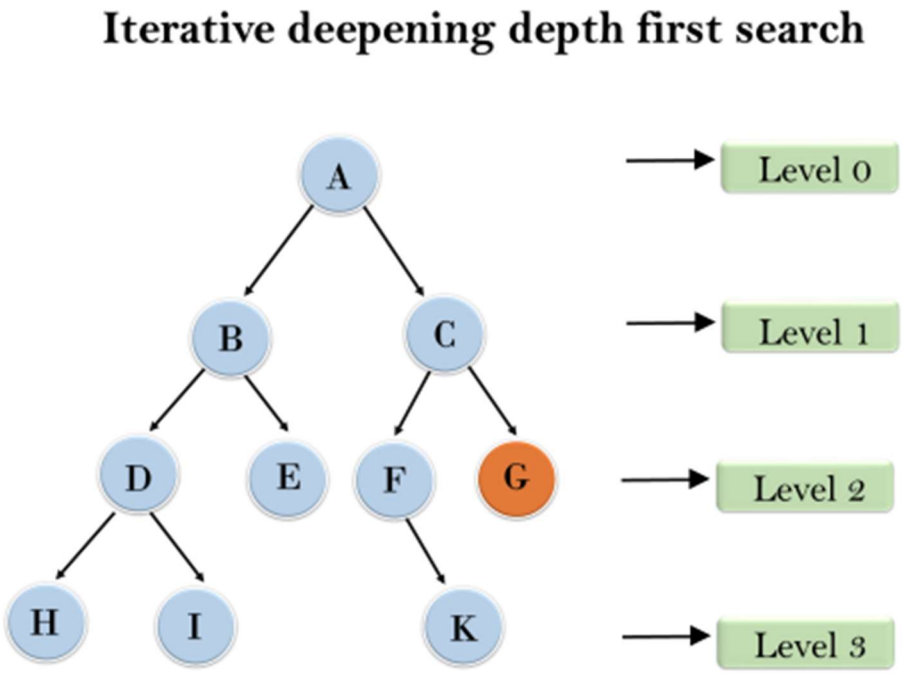
- Itcombines the benefits of BFS and DFS search algorithm in terms of fast search and memory efficiency.

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:



1'st Iteration-----> A
2'nd Iteration----> A, B, C
3'rd Iteration------>A, B, D, E, C, F, G

In the third iteration algorithm will find goal state G

If Goal State would have been F, then it could find it in 4^{th} iteration

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

Completeness:

This algorithm is complete is ifthe 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 $\mathbf{O(b^d)}$.

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.

5. 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 form 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.

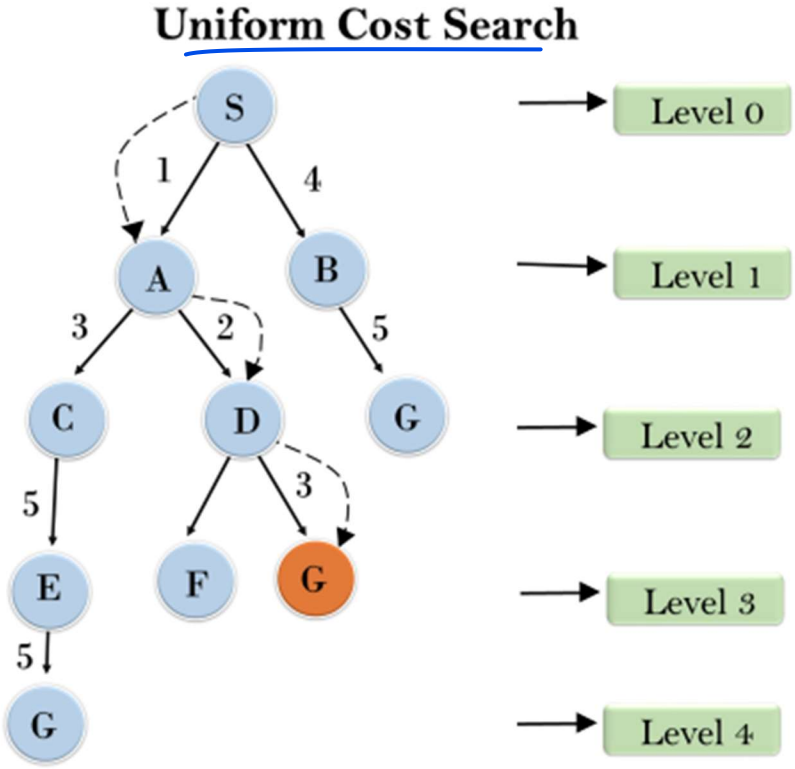
Advantages:

- 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:



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 **ε** is each step to get closer to the goal node. Then the number of steps is = C*/ε+1. Here we have taken +1, as we start from state 0 and end to C*/ε.

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

Space Complexity:

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

Optimal:

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

7. Means-Ends Analysis in Artificial Intelligence

- We have studied the strategies which can reason either in forward or backward, but a mixture of the two directions is appropriate for solving a complex and large problem. Such a mixed strategy, make it possible that first to solve the major part of a problem and then go back and solve the small problems arise during combining the big parts of the problem. Such a technique is called **Means-Ends Analysis**.
- Means-Ends Analysis is problem-solving techniques used in Artificial intelligence for limiting search in AI programs.
- It is a mixture of Backward and forward search technique.
- The MEA technique was first introduced in 1961 by Allen Newell, and Herbert A. Simon in their problem-solving computer program, which was named as General Problem Solver (GPS).
- The MEA analysis process centered on the evaluation of the difference between the current state and goal state.

How means-ends analysis Works:

The means-ends analysis process can be applied recursively for a problem. It is a strategy to control search in problem-solving. Following are the main Steps which describes the working of MEA technique for solving a problem.

a. First, evaluate the difference between Initial State and final State.
b. Select the various operators which can be applied for each difference.
c. Apply the operator at each difference, which reduces the difference between the current state and goal state.

Operator Subgoaling

In the MEA process, we detect the differences between the current state and goal state. Once these differences occur, then we can apply an operator to reduce the differences.

But sometimes it is possible that an operator cannot be applied to the current state. So we create the subproblem of the current state, in which operator can be applied, such type of backward chaining in which operators are selected, and then sub goals are set up to establish the preconditions of the operator is called **Operator Subgoaling**.

Algorithm for Means-Ends Analysis:

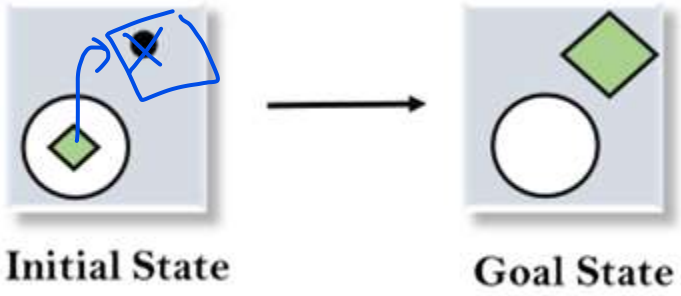
Let's we take Current state as CURRENT and Goal State as GOAL, then following are the steps for the MEA algorithm.

- **Step 1:** Compare CURRENT to GOAL, if there are no differences between both then return Success and Exit.
- **Step 2:** Else, select the most significant difference and reduce it by doing the following steps until the success or failure occurs.
    1. Select a new operator O which is applicable for the current difference, and if there is no such operator, then signal failure.
    2. Attempt to apply operator O to CURRENT. Make a description of two states.
       i) O-Start, a state in which Os preconditions are satisfied.
       ii) O-Result, the state that would result if O were applied In O-start.
    3. If
       (First-Part <------ MEA (CURRENT, O-START)
       And
       (LAST-Part <----- MEA (O-Result, GOAL), are successful, then signal Success and return the result of combining FIRST-PART, O, and LAST-PART.

The above-discussed algorithm is more suitable for a simple problem and not adequate for solving complex problems.

Example of Mean-Ends Analysis:

Let's take an example where we know the initial state and goal state as given below. In this problem, we need to get the goal state by finding differences between the initial state and goal state and applying operators.

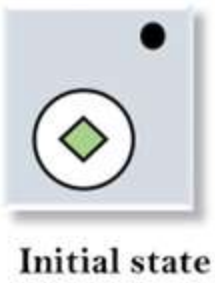


Solution:

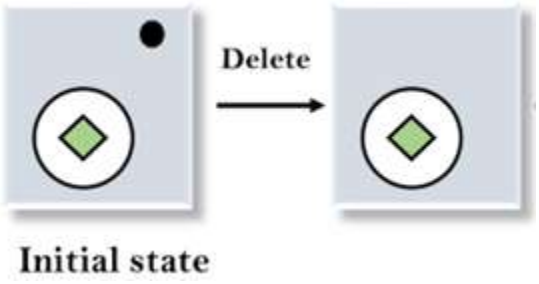
To solve the above problem, we will first find the differences between initial states and goal states, and for each difference, we will generate a new state and will apply the operators. The operators we have for this problem are:

- **Move**
- **Delete**
- **Expand**

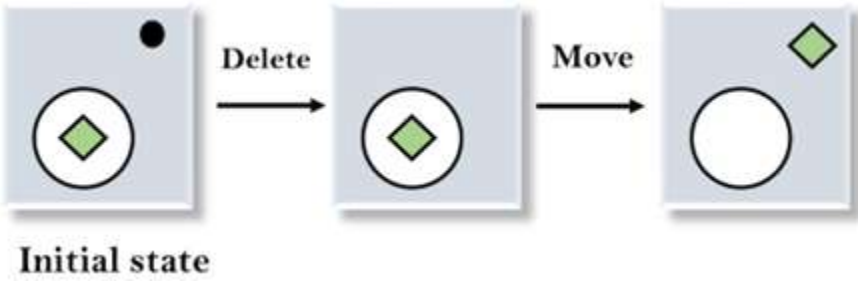
1. Evaluating the initial state: In the first step, we will evaluate the initial state and will compare the initial and Goal state to find the differences between both states.



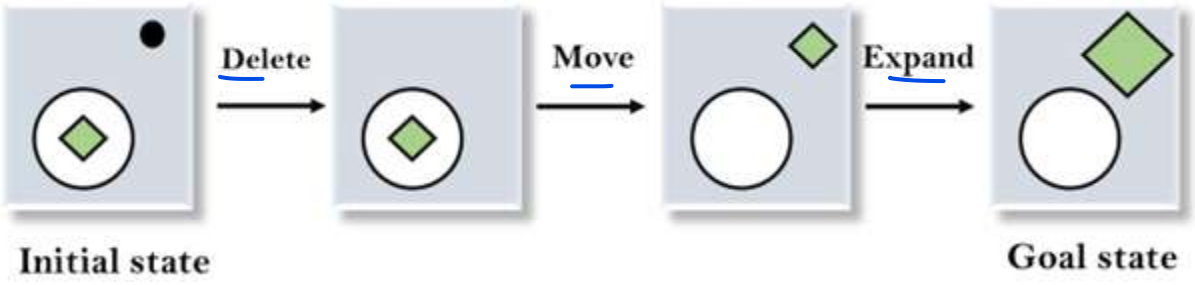
2. Applying Delete operator: As we can check the first difference is that in goal state there is no dot symbol which is present in the initial state, so, first we will apply the **Delete operator** to remove this dot.



3. Applying Move Operator: After applying the Delete operator, the new state occurs which we will again compare with goal state. After comparing these states, there is another difference that is the square is outside the circle, so, we will apply the **Move Operator**.



4. Applying Expand Operator: Now a new state is generated in the third step, and we will compare this state with the goal state. After comparing the states there is still one difference which is the size of the square, so, we will apply **Expand operator**, and finally, it will generate the goal state.



Informed Search Algorithms

So far we have talked about the uninformed search algorithms which looked through search space for all possible solutions of the problem without having any additional knowledge about search space. But informed search algorithm contains an array of knowledge such as how far we are from the goal, path cost, how to reach to goal node, etc. This knowledge help agents to explore less to the search space and find more efficiently the goal node.

The informed search algorithm is more useful for large search space. Informed search algorithm uses the idea of heuristic, so it is also called Heuristic search.

Heuristics function: Heuristic is a function which is used in Informed Search, and it finds the most promising path. It takes the current state of the agent as its input and produces the estimation of how close agent is from the goal. The heuristic method, however, might not always give the best solution, but it guaranteed to find a good solution in reasonable time. Heuristic function estimates how close a state is to the goal. It is represented by h(n), and it calculates the cost of an optimal path between the pair of states. The value of the heuristic function is always positive.

Admissibility of the heuristic function is given as:

1. $h(n) <= h^*(n)$

Here h(n) is heuristic cost, and h*(n) is the estimated cost. Hence heuristic cost should be less than or equal to the estimated cost.

Pure Heuristic Search:

Pure heuristic search is the simplest form of heuristic search algorithms. It expands nodes based on their heuristic value h(n). It maintains two lists, OPEN and CLOSED list. In the CLOSED list, it places those nodes which have already expanded and in the OPEN list, it places nodes which have yet not been expanded.

On each iteration, each node n with the lowest heuristic value is expanded and generates all its successors and n is placed to the closed list. The algorithm continues unit a goal state is found.

-

-
- **Best First Search Algorithm(Greedy search)**
- **A* Search Algorithm**

1.) Best-first Search Algorithm (Greedy Search):

Greedy best-first search algorithm always selects the path which appears best at that moment. It is the combination of depth-first search and breadth-first search algorithms. It uses the heuristic function and search. Best-first search allows us to take the advantages of both algorithms. With the help of best-first search, at each step, we can choose the most promising node. In the best first search algorithm, we expand the node which is closest to the goal node and the closest cost is estimated by heuristic function, i.e.

1. f(n)= g(n).

Were, h(n)= estimated cost from node n to the goal.

The greedy best first algorithm is implemented by the priority queue.

Best first search algorithm:

- **Step 1:** Place the starting node into the OPEN list.
- **Step 2:** If the OPEN list is empty, Stop and return failure.
- **Step 3:** Remove the node n, from the OPEN list which has the lowest value of h(n), and places it in the CLOSED list.
- **Step 4:** Expand the node n, and generate the successors of node n.
- **Step 5:** Check each successor of node n, and find whether any node is a goal node or not. If any successor node is goal node, then return success and terminate the search, else proceed to Step 6.
- **Step 6:** For each successor node, algorithm checks for evaluation function f(n), and then check if the node has been in either OPEN or CLOSED list. If the node has not been in both list, then add it to the OPEN list.
- **Step 7:** Return to Step 2.

Advantages:

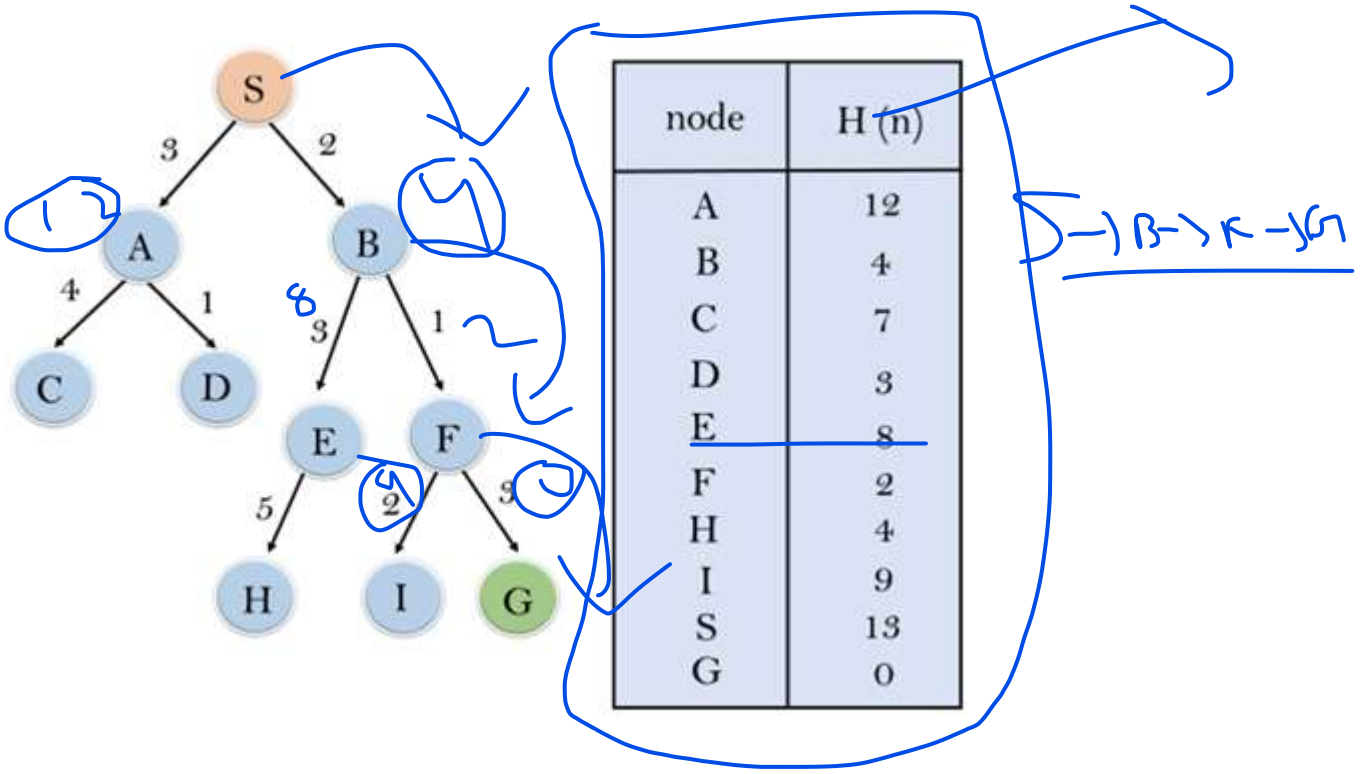
- Best first search can switch between BFS and DFS by gaining the advantages of both the algorithms.
- This algorithm is more efficient than BFS and DFS algorithms.

Disadvantages:

- It can behave as an unguided depth-first search in the worst case scenario.
- It can get stuck in a loop as DFS.
- This algorithm is not optimal.

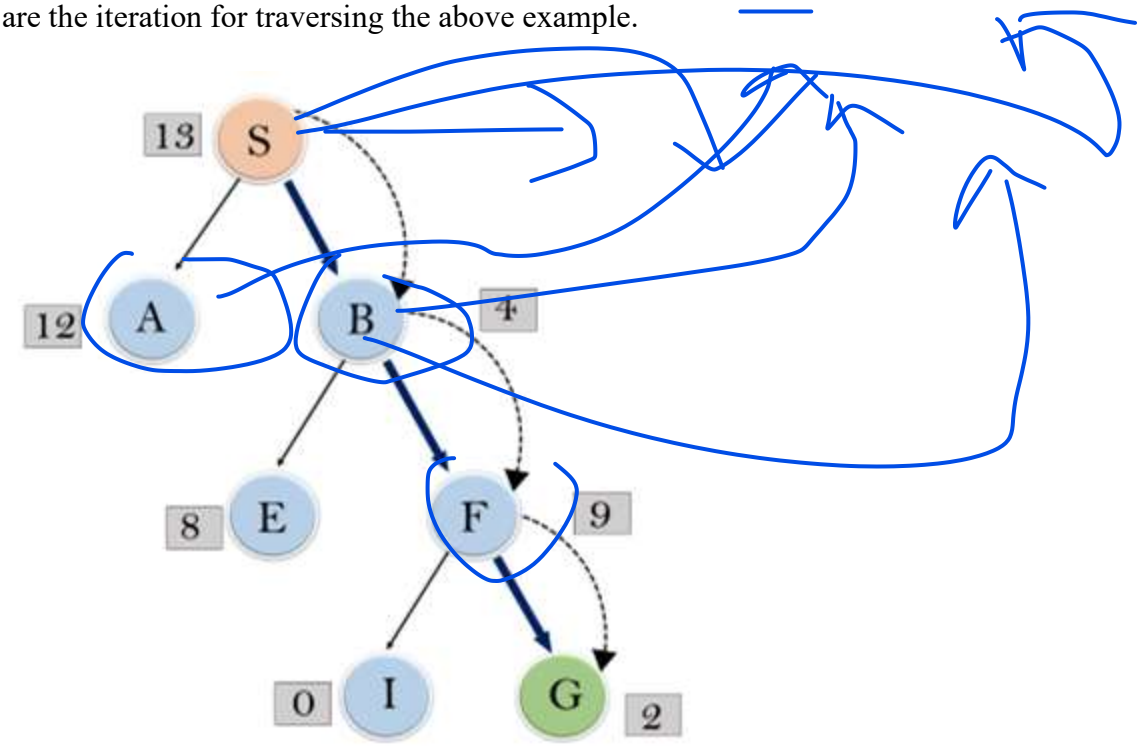
Example:

Consider the below search problem, and we will traverse it using greedy best-first search. At each iteration, each node is expanded using evaluation function f(n)=h(n) , which is given in the below table.



node	H (n)
A	12
B	4
C	7
D	3
E	8
F	2
H	4
I	9
S	13
G	0

In this search example, we are using two lists which are **OPEN** and **CLOSED** Lists. Following are the iteration for traversing the above example.



Expand the nodes of S and put in the CLOSED list

Initialization: Open [A, B], Closed [S]

Iteration 1: Open [A], Closed [S, B]

Iteration 2: Open [E, F, A], Closed [S, B]
: Open [E, A], Closed [S, B, F]

Iteration 3: Open [I, G, E, A], Closed [S, B, F]
: Open [I, E, A], Closed [S, B, F, G]

Hence the final solution path will be: **S----> B----->F----> G**

Time Complexity: The worst case time complexity of Greedy best first search is $O(b^m)$.

Space Complexity: The worst case space complexity of Greedy best first search is $O(b^m)$. Where, m is the maximum depth of the search space.

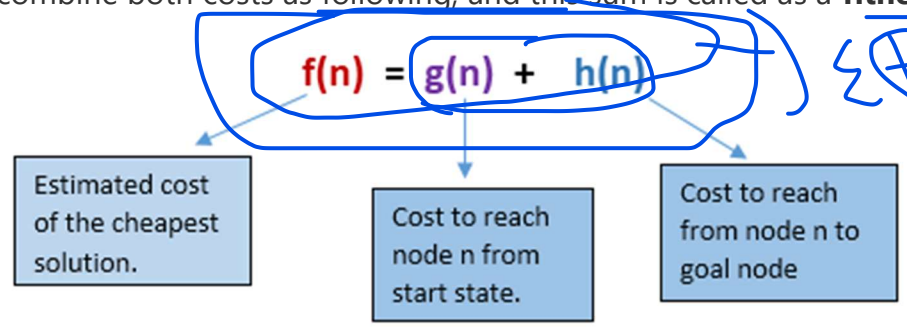
Complete: Greedy best-first search is also incomplete, even if the given state space is finite.

Optimal: Greedy best first search algorithm is not optimal.

2.) A* Search Algorithm:

A* search is the most commonly known form of best-first search. It uses heuristic function h(n), and cost to reach the node n from the start state g(n). It has combined features of UCS and greedy best-first search, by which it solve the problem efficiently. A* search algorithm finds the shortest path through the search space using the heuristic function. This search algorithm expands less search tree and provides optimal result faster. A* algorithm is similar to UCS except that it uses g(n)+h(n) instead of g(n).

In A* search algorithm, we use search heuristic as well as the cost to reach the node. Hence we can combine both costs as following, and this sum is called as a **fitness number**.



At each point in the search space, only those nodes are expanded which have the lowest value of f(n), and the algorithm terminates when the goal node is found.

Algorithm of A* search:

Step1: Place the starting node in the OPEN list.

Step 2: Check if the OPEN list is empty or not, if the list is empty then return failure and stops.

Step 3: Select the node from the OPEN list which has the smallest value of evaluation function (g+h), if node n is goal node then return success and stop, otherwise

Step 4: Expand node n and generate all of its successors, and put n into the closed list. For each successor n', check whether n' is already in the OPEN or CLOSED list, if not then compute evaluation function for n' and place into Open list.

Step 5: Else if node n' is already in OPEN and CLOSED, then it should be attached to the back pointer which reflects the lowest g(n') value.

Step 6: Return to **Step 2**.

Advantages:

- A* search algorithm is the best algorithm than other search algorithms.
- A* search algorithm is optimal and complete.
- This algorithm can solve very complex problems.

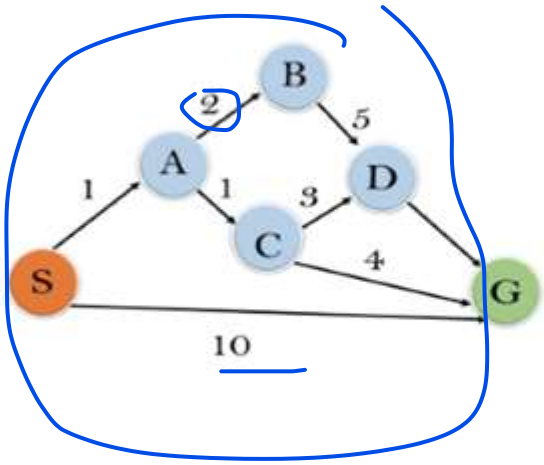
Disadvantages:

- It does not always produce the shortest path as it mostly based on heuristics and approximation.
- A* search algorithm has some complexity issues.
- The main drawback of A* is memory requirement as it keeps all generated nodes in the memory, so it is not practical for various large-scale problems.

Example:

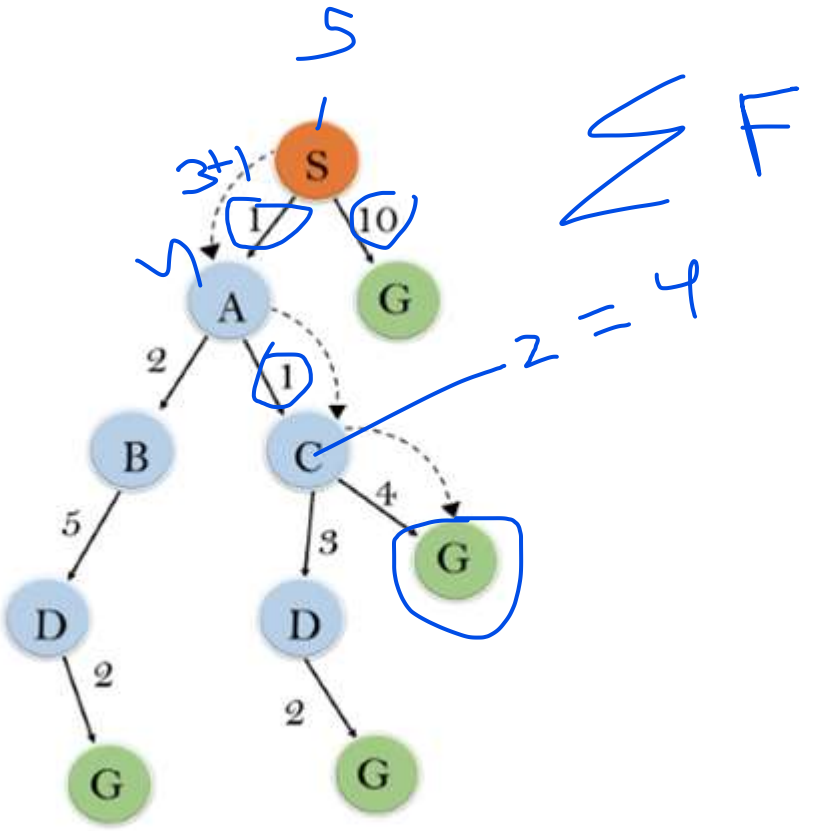
In this example, we will traverse the given graph using the A* algorithm. The heuristic value of all states is given in the below table so we will calculate the f(n) of each state using the formula f(n)= g(n) + h(n), where g(n) is the cost to reach any node from start state.

Here we will use OPEN and CLOSED list.



State	h(n)
S	5
A	3
B	4
C	2
D	6
G	0

Solution:



Initialization: {(S, 5)}

Iteration1: {(S--> A, 4), (S-->G, 10)}

Iteration2: {(S--> A-->C, 4), (S--> A-->B, 7), (S-->G, 10)}

Iteration3: {(S--> A-->C--->G, 6), (S--> A-->C--->D, 11), (S--> A-->B, 7), (S-->G, 10)}

Iteration 4 will give the final result, as **S--->A--->C--->G** it provides the optimal path with cost 6.

Points to remember:

- A* algorithm returns the path which occurred first, and it does not search for all remaining paths.
- The efficiency of A* algorithm depends on the quality of heuristic.
- A* algorithm expands all nodes which satisfy the condition f(n)<="" li="">

Complete: A* algorithm is complete as long as:

- Branching factor is finite.
- Cost at every action is fixed.

Optimal: A* search algorithm is optimal if it follows below two conditions:

- **Admissible:** the first condition requires for optimality is that h(n) should be an admissible heuristic for A* tree search. An admissible heuristic is optimistic in nature.
- **Consistency:** Second required condition is consistency for only A* graph-search.

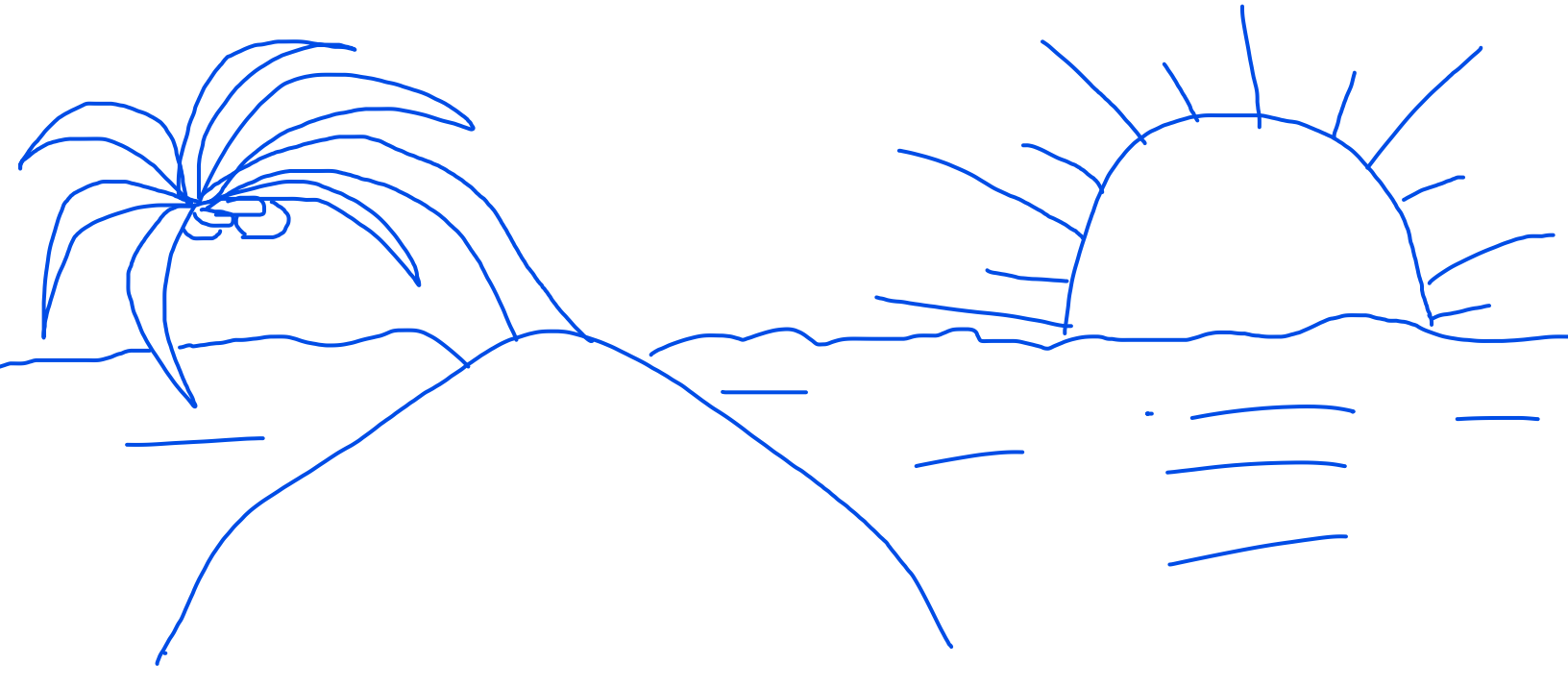
If the heuristic function is admissible, then A* tree search will always find the least cost path.

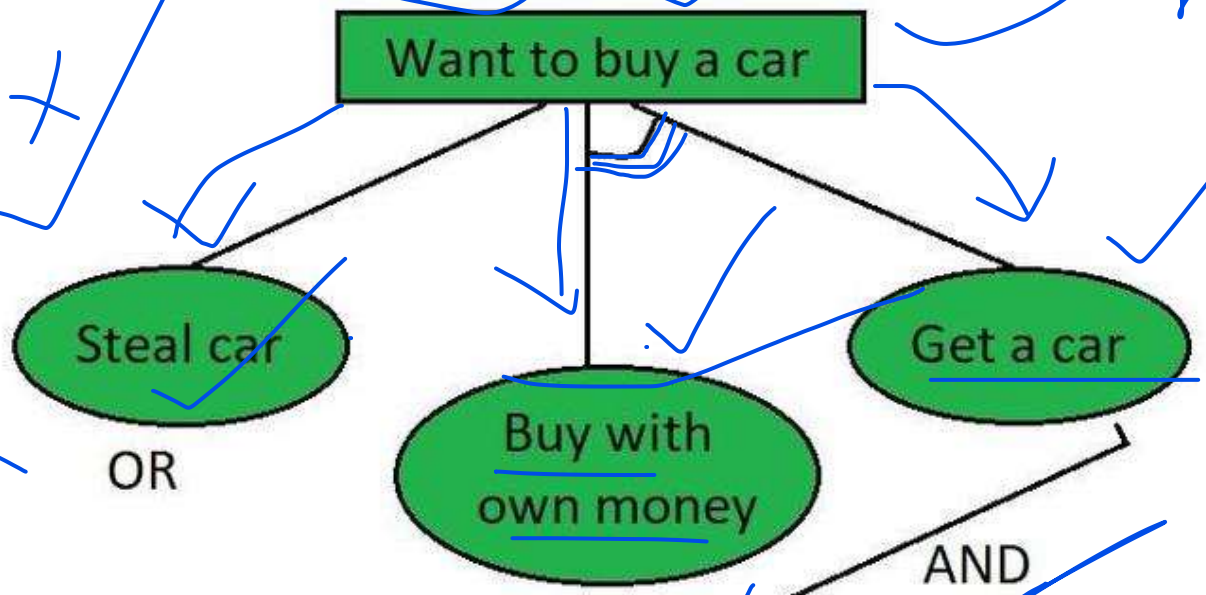
Time Complexity: The time complexity of A* search algorithm depends on heuristic function, and the number of nodes expanded is exponential to the depth of solution d. So the time complexity is O(b^d), where b is the branching factor.

Space Complexity: The space complexity of A* search algorithm is **O(b^d)**

AO* algorithm – Artificial intelligence

Best-first search is what the AO* algorithm does. The AO* method **divides** any given difficult **problem into a smaller group** of problems that are then resolved **using the AND-OR** graph concept. AND OR graphs are specialized graphs that are used in problems that can be divided into smaller problems. The AND side of the graph represents a set of tasks that must be completed to achieve the main goal, while the OR side of the graph represents different methods for accomplishing the same main goal.





AND-OR Graph

In the above figure, the **buying of a car** may be broken down into smaller problems or tasks that can be accomplished **to achieve the main goal** in the above figure, which is an example of a simple AND-OR graph. The other task is to either steal a car that will help us accomplish the main goal or use your own money to purchase a car that will accomplish the main goal. The AND symbol is used to indicate the AND part of the graphs, which refers to the need that all subproblems containing the AND to be resolved before the preceding node or issue may be finished.

The start state and the target state are already known in the knowledge-based search strategy known as the **AO* algorithm**, and the best path is identified by heuristics. The informed search technique considerably reduces the algorithm's **time complexity**. The AO* algorithm is far more effective in searching AND-OR trees **than** the A* algorithm.

Working of AO* algorithm:

The evaluation function in AO* looks like this:

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

f(n) = Actual cost + Estimated cost

here,

f(n) = The actual cost of traversal.

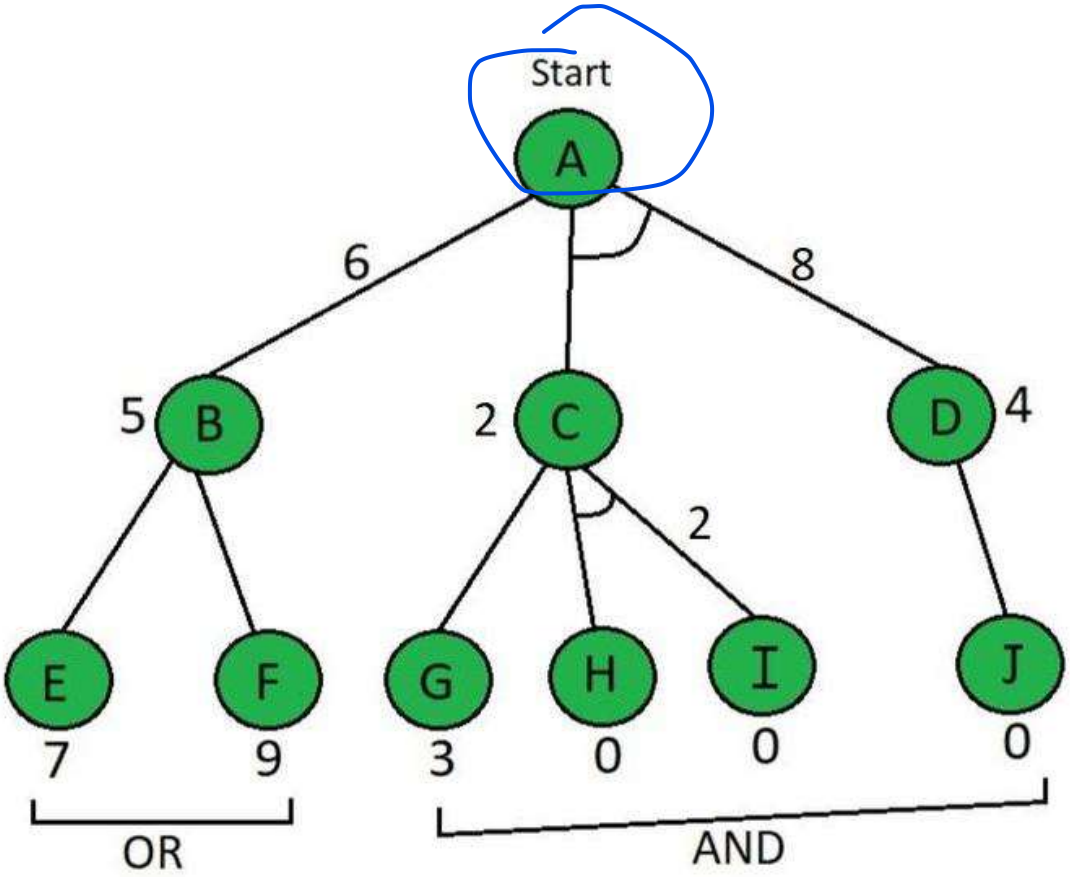
g(n) = the cost from the initial node to the current node.

h(n) = estimated cost from the current node to the goal state.

Difference between the A* Algorithm and AO* algorithm

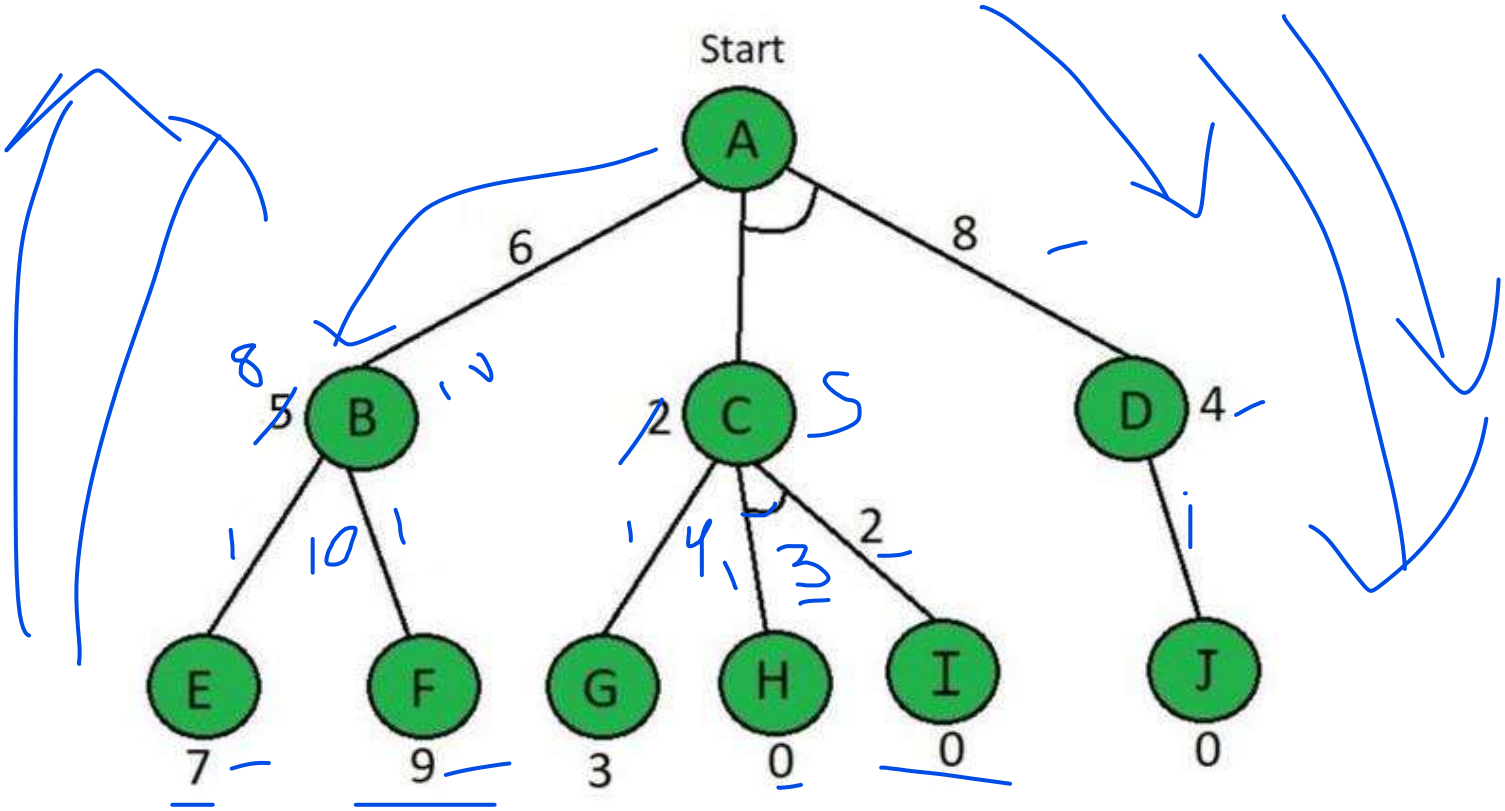
- A* algorithm and AO* algorithm both works on the **best first search**.
- They are both **informed search** and works on given heuristics values.
- **A*** always **gives** the **optimal solution** but AO* doesn't guarantee to give the optimal solution.
- Once AO* got a solution **doesn't explore** all possible paths but A* explores all paths.
- When compared to the A* algorithm, the AO* algorithm uses **less memory.**
- opposite to the A* algorithm, the AO* algorithm cannot go into an endless **loop.**

Example:



AO Algorithm – Question tree*

Here in the above example below the Node which is given is the heuristic value i.e **h(n)**. Edge length is considered as **1**.
Step 1



AO Algorithm (Step-1)*

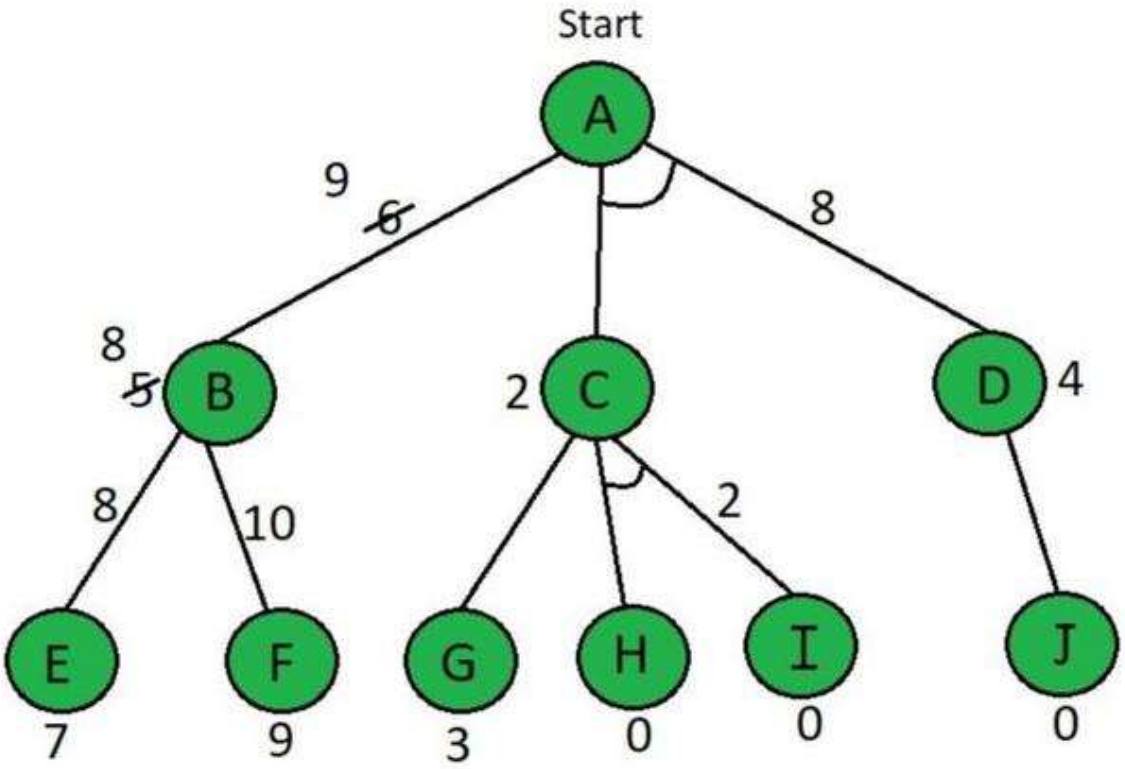
With help of **f(n) = g(n) + h(n)** evaluation function,
Start from node A,

f(A⇢B) = g(B) + h(B)
= 1 + 5 .....here **g(n)=1** is taken by default for path cost
= 6

f(A⇢C+D) = g(c) + h(c) + g(d) + h(d)
= 1 + 2 + 1 + 4 .....here we have added **C & D** because they are in **AND**
= 8

So, by calculation **A⇢B** path is chosen which is the minimum path, i.e **f(A⇢B)**

Step 2



AO Algorithm (Step-2)*

According to the answer of step 1, explore node B

Here the value of E & F are calculated as follows,

f(B⇢E) = g(e) + h(e)

f(B⇢E) = 1 + 7

= 8

f(B⇢f) = g(f) + h(f)

f(B⇢f) = 1 + 9

= 10

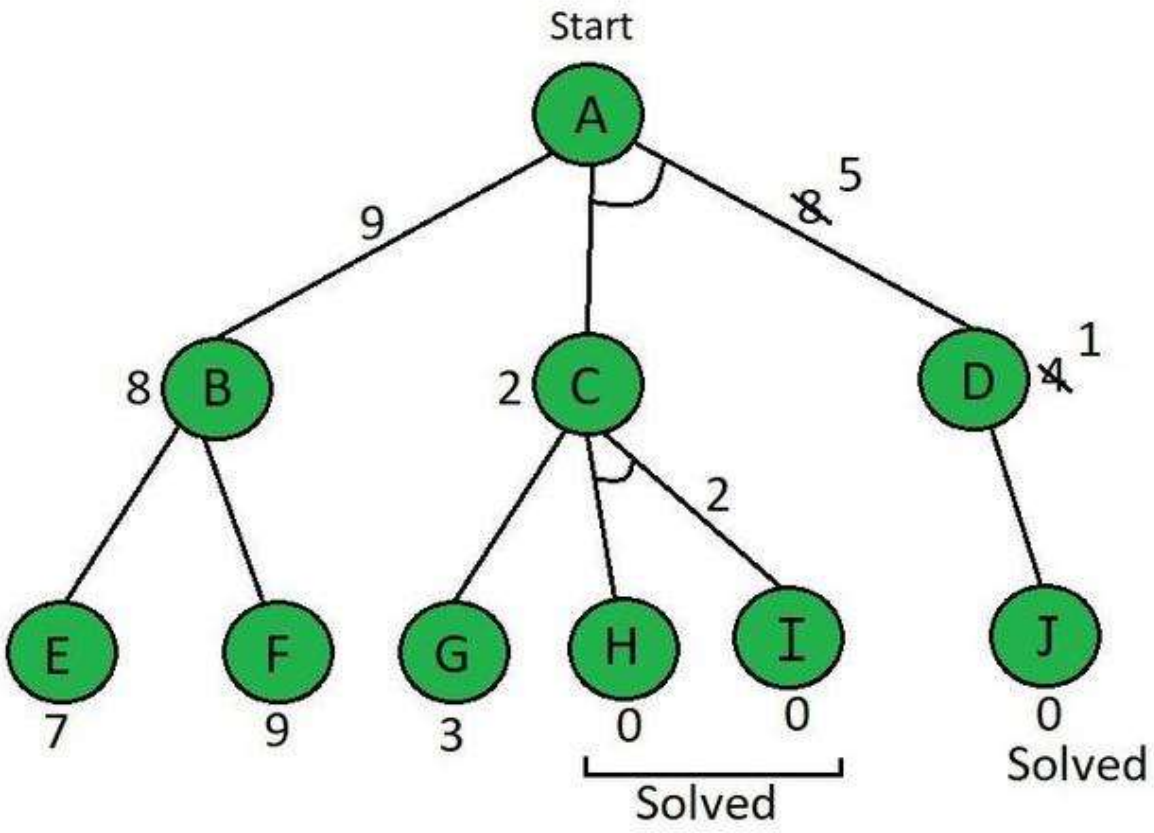
So, by above calculation B⇢E path is chosen which is minimum path, i.e **f(B⇢E)**

because **B's** heuristic value is different from its actual value The heuristic is
updated and the minimum cost path is selected. The minimum value in our situation is **8.**
Therefore, the heuristic for **A** must be updated due to the change in **B's** heuristic.
So we need to calculate it again.

```
f(A⇢B) = g(B) + updated h(B)
       = 1 + 8
       = 9
```

We have Updated all values in the above tree.

Step 3



AO Algorithm (Step-3) -Geeksforgeeks*

By comparing **f(A⇢B)** & **f(A⇢C+D)**
f(A⇢C+D) is shown to be **smaller.** i.e 8 < 9
Now explore f(A⇢C+D)

So, the current node is **C**

f(C⇢G) = g(g) + h(g)
f(C⇢G) = 1 + 3
= 4

f(C⇢H+I) = g(h) + h(h) + g(i) + h(i)
f(C⇢H+I) = 1 + 0 + 1 + 0 ……here we have added **H & I** because they are in **AND**
= 2

f(C⇢H+I) is selected as the path with the lowest cost and the heuristic is also left unchanged
because it matches the actual cost. Paths H & I are solved because the heuristic for those paths is **0,**
but Path **A⇢D** needs to be calculated because it has an **AND.**

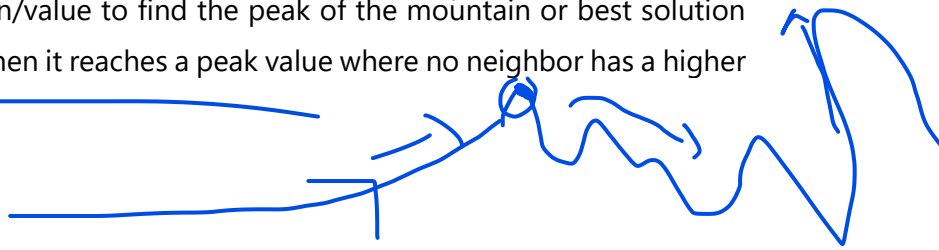
f(D⇢J) = g(j) + h(j)
f(D⇢J) = 1 + 0
= 1
the heuristic of node D needs to be updated to 1.

f(A⇢C+D) = g(c) + h(c) + g(d) + h(d)
= 1 + 2 + 1 + 1
= 5

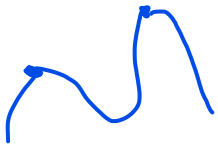
as we can see that path **f(A⇢C+D)** is get solved and this **tree has become a solved tree** now.
In simple words, the main flow of this algorithm is that we have to find **firstly level 1st** heuristic
value and **then level 2nd** and after that **update the values** with going **upward** means towards the root node.
In the above tree diagram, we have updated all the values.

Hill Climbing Algorithm in Artificial Intelligence

- Hill climbing algorithm is a local search algorithm which continuously moves in the direction of increasing elevation/value to find the peak of the mountain or best solution to the problem. It terminates when it reaches a peak value where no neighbor has a higher value.



- Hill climbing algorithm is a technique which is used for optimizing the mathematical problems. One of the widely discussed examples of Hill climbing algorithm is Traveling-salesman Problem in which we need to minimize the distance traveled by the salesman.
- It is also called greedy local search as it only looks to its good immediate neighbor state and not beyond that.
- A node of hill climbing algorithm has two components which are state and value.
- Hill Climbing is mostly used when a good heuristic is available.
- In this algorithm, we don't need to maintain and handle the search tree or graph as it only keeps a single current state.
-



Features of Hill Climbing:

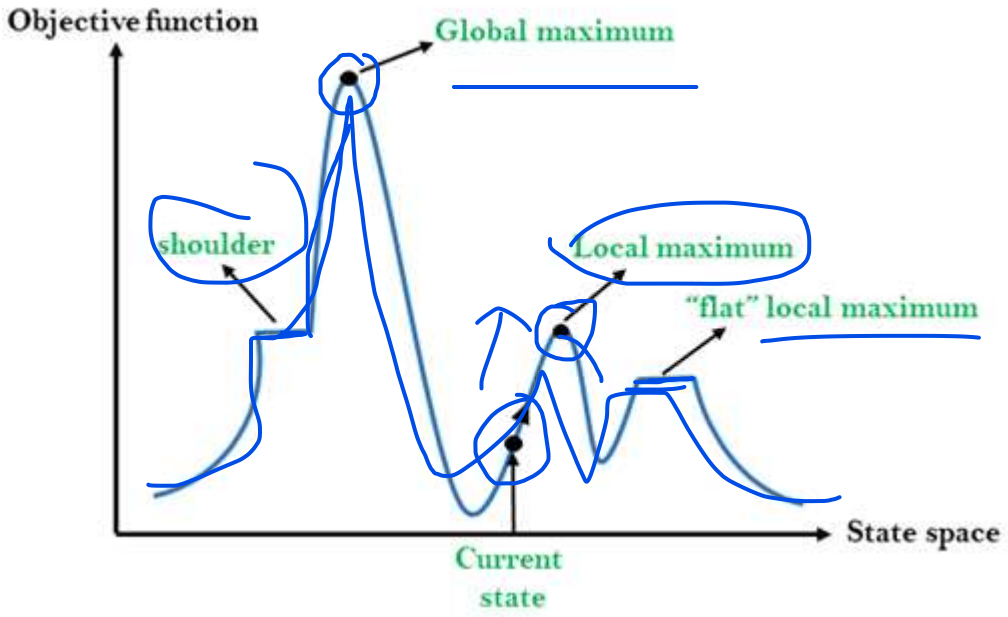
Following are some main features of Hill Climbing Algorithm:

- **Generate and Test variant:** Hill Climbing is the variant of Generate and Test method. The Generate and Test method produce feedback which helps to decide which direction to move in the search space.
- **Greedy approach:** Hill-climbing algorithm search moves in the direction which optimizes the cost.
- **No backtracking:** It does not backtrack the search space, as it does not remember the previous states.

State-space Diagram for Hill Climbing:

The state-space landscape is a graphical representation of the hill-climbing algorithm which is showing a graph between various states of algorithm and Objective function/Cost.

On Y-axis we have taken the function which can be an objective function or cost function, and state-space on the x-axis. If the function on Y-axis is cost then, the goal of search is to find the global minimum and local minimum. If the function of Y-axis is Objective function, then the goal of the search is to find the global maximum and local maximum.



Different regions in the state space landscape:

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

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

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

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

Shoulder: It is a plateau region which has an uphill edge.

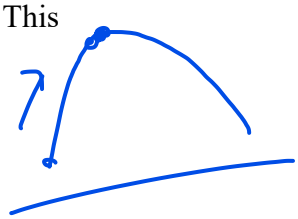
Types of Hill Climbing Algorithm:

- Simple hill Climbing:
- Steepest-Ascent hill-climbing:
- Stochastic hill Climbing:

1. Simple Hill Climbing:

Simple hill climbing is the simplest way to implement a hill climbing algorithm. **It only evaluates the neighbor node state at a time and selects the first one which**

optimizes current cost and set it as a current state. It only checks it's one successor state, and if it finds better than the current state, then move else be in the same state. This algorithm has the following features:



- Less time consuming
- Less optimal solution and the solution is not guaranteed

Algorithm for Simple Hill Climbing:

- **Step 1:** Evaluate the initial state, if it is goal state then return success and Stop.
- **Step 2:** Loop Until a solution is found or there is no new operator left to apply.
- **Step 3:** Select and apply an operator to the current state.
- **Step 4:** Check new state:
    1. If it is goal state, then return success and quit.
    2. Else if it is better than the current state then assign new state as a current state.
    3. Else if not better than the current state, then return to step2.
- **Step 5:** Exit.

2. Steepest-Ascent hill climbing:

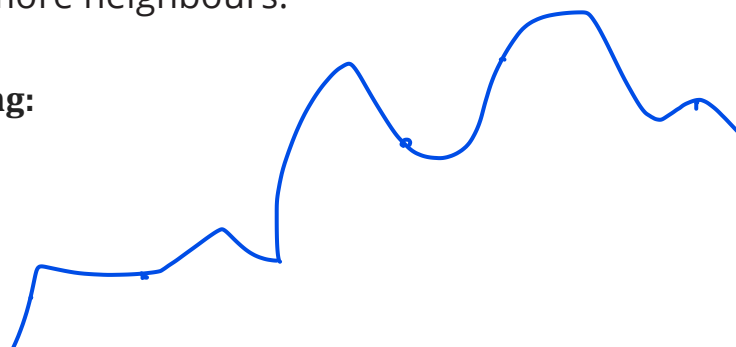
The steepest-Ascent algorithm is a variation of simple hill climbing algorithm. This algorithm examines all the neighboring nodes of the current state and selects one neighbor node which is closest to the goal state. This algorithm consumes more time as it searches for multiple neighbors

3. Stochastic hill climbing:

Stochastic hill climbing does not examine for all its neighbor before moving. Rather, this search algorithm selects one neighbor node at random and decides whether to choose it as a current state or examine another state.

Steepest-Ascent Hill Climbing

A variant of the straightforward hill-climbing algorithm is the steepest-Ascent algorithm. This method looks at every node that borders the current state and chooses the one that is most near the goal state. This algorithm takes longer since it looks for more neighbours.



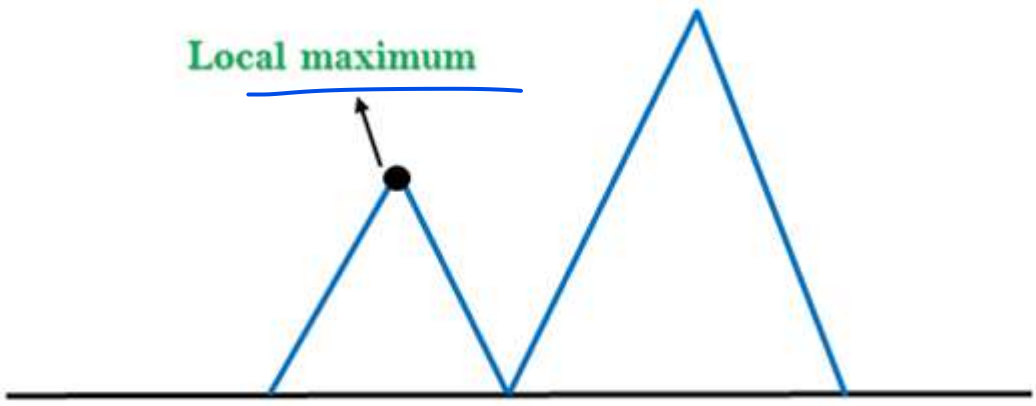
Algorithm for Steepest Ascent Hill climbing:

- Analyze the starting situation. Stop and return success if it's a goal state. If not, the initial state should be set as the current state.
- Follow these instructions again and again until a solution is found, or the situation stays the same.
- Choose a state that hasn't yet been used to modify the existing state.
- Create a new "best state" that is initially equivalent to the existing state and then apply it to create the new state.
- Execute these to assess the new state.
- Stop and return success if the current state is a goal state.
- If it is superior to the best state, make it the best state; otherwise, keep going by adding another new state to the loop.
- Set the ideal situation as the current situation.
- Exit

Problems in Hill Climbing Algorithm:

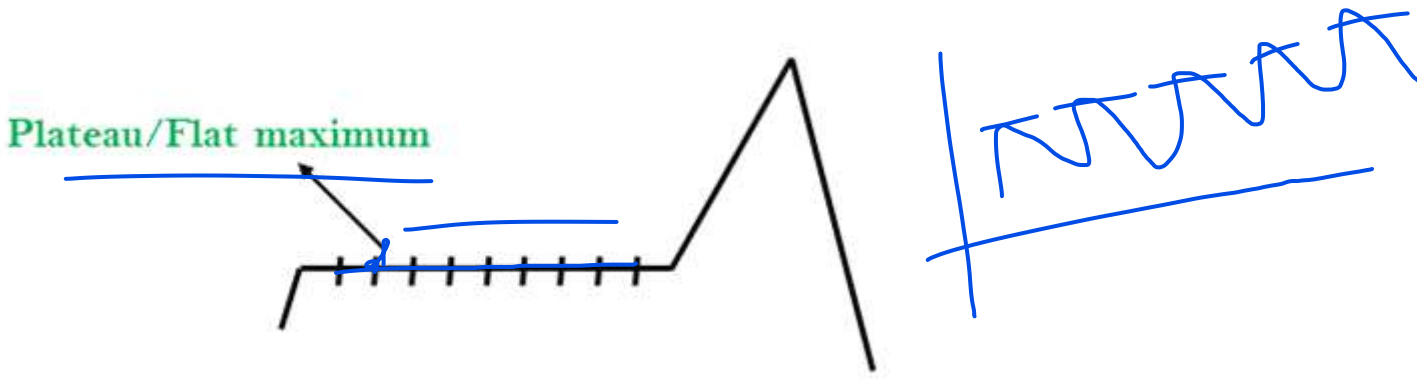
1. Local Maximum: A local maximum is a peak state in the landscape which is better than each of its neighboring states, but there is another state also present which is higher than the local maximum.

Solution: Backtracking technique can be a solution of the local maximum in state space landscape. Create a list of the promising path so that the algorithm can backtrack the search space and explore other paths as well.



2. Plateau: A plateau is the flat area of the search space in which all the neighbor states of the current state contains the same value, because of this algorithm does not find any best direction to move. A hill-climbing search might be lost in the plateau area.

Solution: The solution for the plateau is to take big steps or very little steps while searching, to solve the problem. Randomly select a state which is far away from the current state so it is possible that the algorithm could find non-plateau region.



3.Ridges: Any point on a ridge can appear as a peak since all directions of movement are downhill. As a result, the algorithm terminates in this condition.

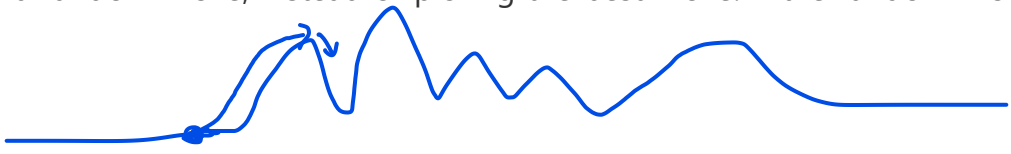
To get over a Ridge: follow two or more rules before being tested. It suggests acting simultaneously in numerous directions.

> **Solution:** With the use of bidirectional search, or by moving in different directions, we can improve this problem.

Simulated Annealing:

A hill-climbing algorithm which never makes a move towards a lower value guaranteed to be incomplete because it can get stuck on a local maximum. And if algorithm applies a random walk, by moving a successor, then it may complete but not efficient. **Simulated Annealing** is an algorithm which yields both efficiency and completeness.

In mechanical term **Annealing** is a process of hardening a metal or glass to a high temperature then cooling gradually, so this allows the metal to reach a low-energy crystalline state. The same process is used in simulated annealing in which the algorithm picks a random move, instead of picking the best move. If the random move improves



the state, then it follows the same path. Otherwise, the algorithm follows the path which has a probability of less than 1 or it moves downhill and chooses another path.