Artificial Intelligence Tutorial



The Artificial Intelligence tutorial provides an introduction to AI which will help you to understand the concepts behind [Artificial Intelligence](https://www.tutorialandexample.com/artificial-intelligence-tutorial/). In this tutorial, we have also discussed various popular topics such as History of AI, applications of AI, deep learning, machine learning, natural language processing, Reinforcement learning, Q-learning, Intelligent agents, Various search algorithms, etc.

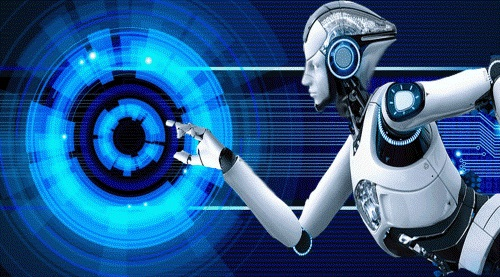
Our AI tutorial is prepared from an elementary level so you can easily understand the complete tutorial from basic concepts to the high-level concepts.

What is Artificial Intelligence?

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:

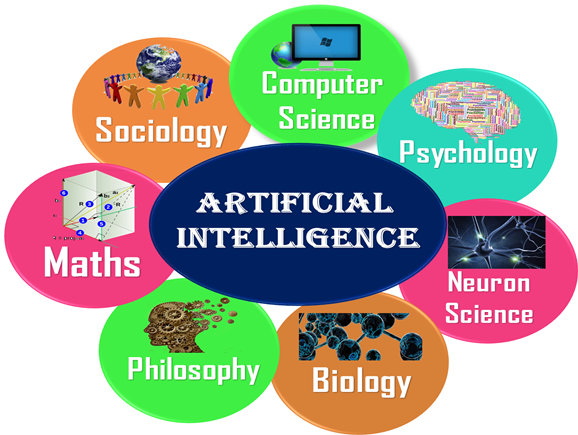
1. Replicate human intelligence
2. Solve Knowledge-intensive tasks
3. An intelligent connection of perception and action
4. Building a machine which can perform tasks that requires human intelligence such as:
   * Proving a theorem
   * Playing chess
   * Plan some surgical operation
   * Driving a car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

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.

Prerequisite

Before learning about Artificial Intelligence, you must have the fundamental knowledge of following so that you can understand the concepts easily:

* Any computer language such as C, C++, Java, Python, etc.(knowledge of Python will be an advantage)
* Knowledge of essential Mathematics such as derivatives, probability theory, etc.

Audience

Our AI tutorial is designed specifically for beginners and also included some high-level concepts for professionals.

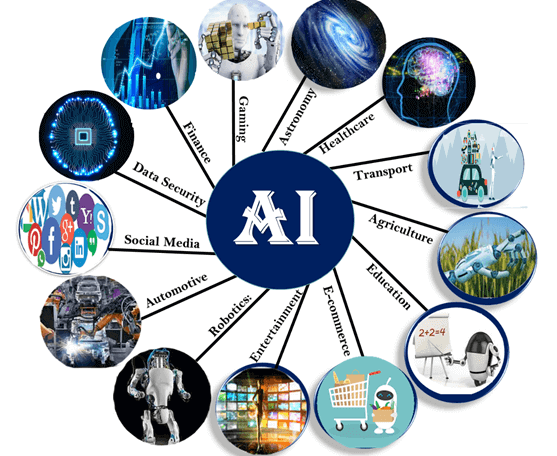
Problems

We assure you that you will not find any difficulty while learning our AI tutorial. But if there any mistake, kindly post the problem in the contact form.

# 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**

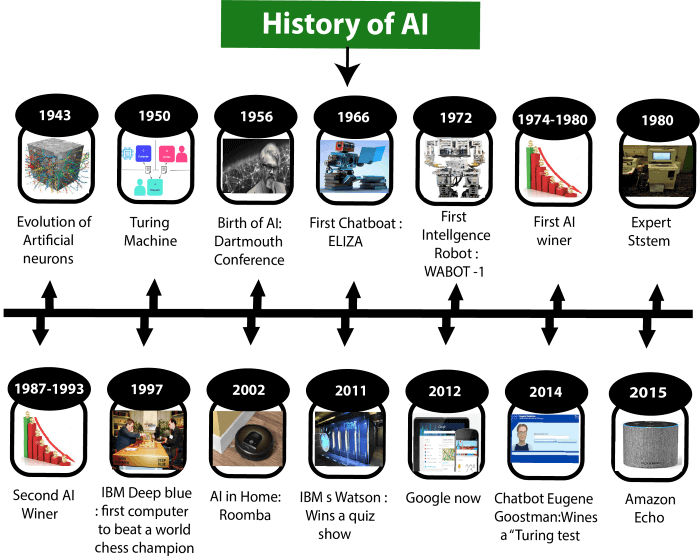
* AI is providing a competitive edge to the e-commerce industry, and it is becoming more demanding in the e-commerce business. AI is helping shoppers to discover associated products with recommended size, color, or even brand.

### **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**.
* **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.

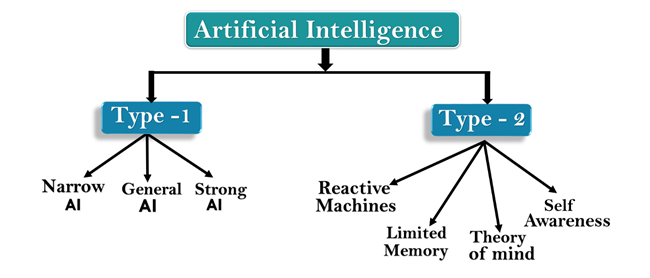
Deep learning, big data and artificial general intelligence (2011-present)

* **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:**

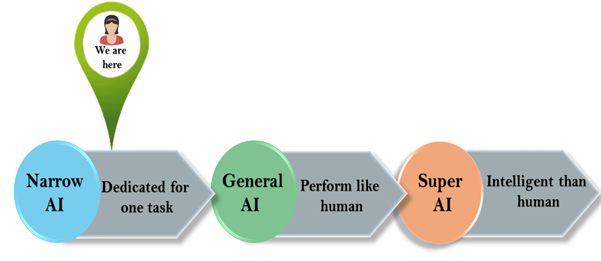
* Narrow AI is a type of AI which is able to perform a dedicated task with intelligence.The most common and currently available AI is Narrow AI in the world of Artificial Intelligence.
* Narrow AI cannot perform beyond its field or limitations, as it is only trained for one specific task. Hence it is also termed as weak AI. Narrow AI can fail in unpredictable ways if it goes beyond its limits.
* Apple Siriis 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 is an example of reactive machines.
* Google's AlphaGo 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.

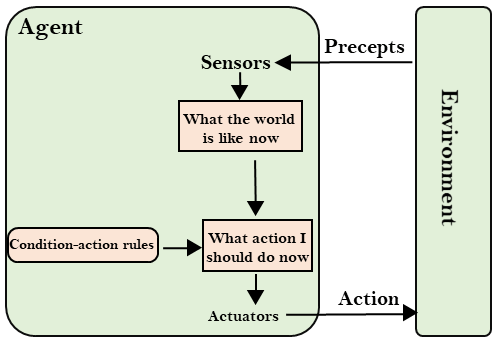
Types of AI Agents

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

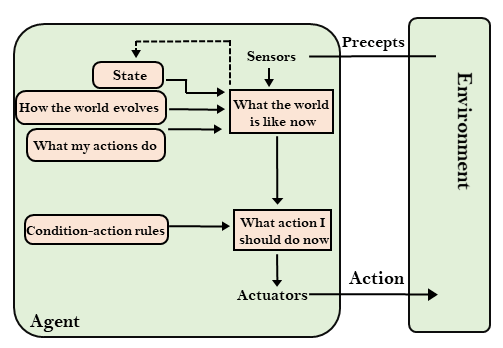
1. Simple Reflex agent:

* 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.



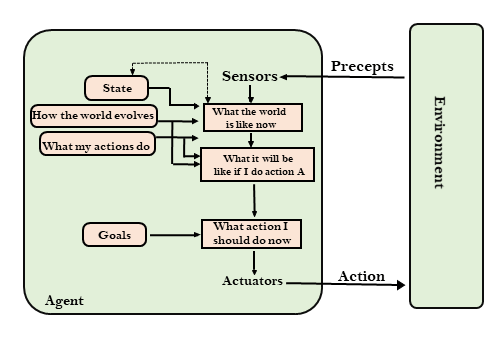
2. Model-based reflex agent

* 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.



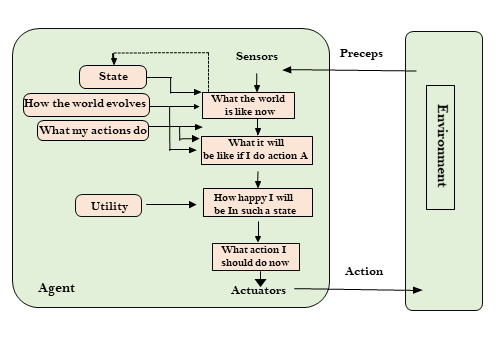
3. Goal-based agents

* 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.



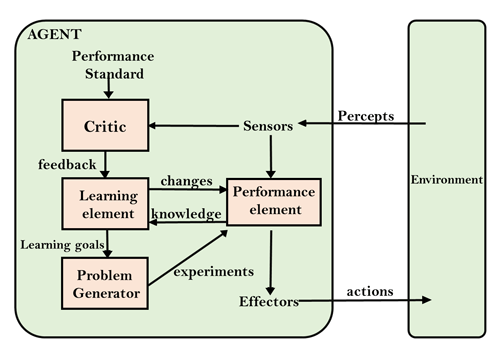
4. Utility-based agents

* 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.



# 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 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.

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

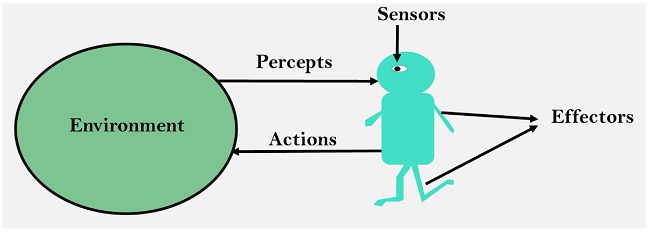
ADVERTISING

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.

#### **Note: Rational agents in AI are very similar to intelligent agents.**

### **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.

#### **Note: Rationality differs from Omniscience because an Omniscient agent knows the actual outcome of its action and act accordingly, which is not possible in reality.**

## 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\* → A

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

## 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 sensor * Cliff sensor * Bump Sensor * Infrared Wall Sensor |
| **3. Part -picking Robot** | * Percentage of parts in correct bins. | * Conveyor belt with parts, * Bins | * Jointed Arms * Hand | * Camera * Joint angle sensors. |

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. An environment is mostly said to be non-feministic.

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.

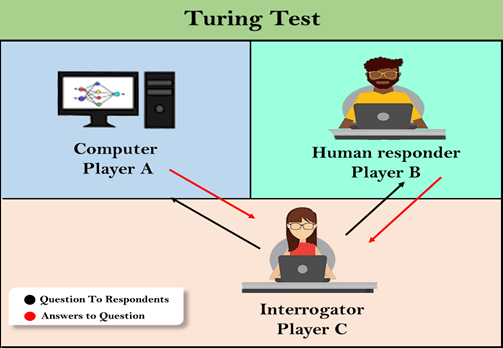
8. Accessible vs Inaccessible

* 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.

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.

The questions and answers can be like:

**Interrogator:** Are you a computer?

**PlayerA (Computer):** No

**Interrogator:** Multiply two large numbers such as (256896489\*456725896)

**Player A:** Long pause and give the wrong answer.

In this game, if an interrogator would not be able to identify which is a machine and which is human, then the computer passes the test successfully, and the machine is said to be intelligent and can think like a human.

"In 1991, the New York businessman Hugh Loebner announces the prize competition, offering a $100,000 prize for the first computer to pass the Turing test. However, no AI program to till date, come close to passing an undiluted Turing test".

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.

The Chinese Room Argument:

There were many philosophers who really disagreed with the complete concept of Artificial Intelligence. The most famous argument in this list was "**Chinese Room**."

In the year **1980, John Searle** presented "**Chinese Room**" thought experiment, in his paper "**Mind, Brains, and Program**," which was against the validity of Turing's Test. According to his argument, "**Programming a computer may make it to understand a language, but it will not produce a real understanding of language or consciousness in a computer**."

He argued that Machine such as ELIZA and Parry could easily pass the Turing test by manipulating keywords and symbol, but they had no real understanding of language. So it cannot be described as "thinking" capability of a machine such as a human.

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.

# Search Algorithms in Artificial Intelligence

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.

## Search Algorithm Terminologies:

* **Search:** Searchingis a step by step procedure to solve a search-problem in a given search space. A search problem can have three main factors:
  1. **Search Space:** Search space represents a set of possible solutions, which a system may have.
  2. **Start State:** It is a state from where agent begins **the search**.
  3. **Goal test:** It is a function which observe the current state and returns whether the goal state is achieved or not.
* **Search tree:** A tree representation of search problem is called Search tree. The root of the search tree is the root node which is corresponding to the initial state.
* **Actions:** It gives the description of all the available actions to the agent.
* **Transition model:** A description of what each action do, can be represented as a transition model.
* **Path Cost:** It is a function which assigns a numeric cost to each path.
* **Solution:** It is an action sequence which leads from the start node to the goal node.
* **Optimal Solution:** If a solution has the lowest cost among all solutions.

## Properties of Search Algorithms:

Following are the four essential properties of search algorithms to compare the efficiency of these algorithms:

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

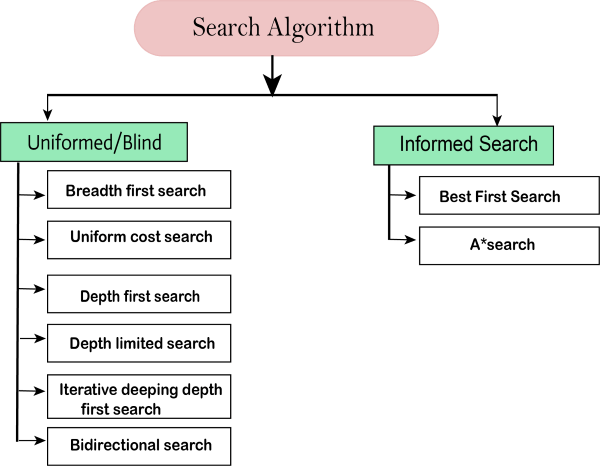
**Optimality:** If a solution found for an algorithm is guaranteed to be the best solution (lowest path cost) among all other solutions, then such a solution for is said to be an optimal solution.

**Time Complexity:** Time complexity is a measure of time for an algorithm to complete its task.

**Space Complexity:** It is the maximum storage space required at any point during the search, as the complexity of the problem.

## Types of search algorithms

**Based on the search problems we can classify the search algorithms into uninformed (Blind search) search and informed search (Heuristic search) algorithms.**



### **Uninformed/Blind Search:**

The uninformed search does not contain any domain knowledge such as closeness, the location of the goal. It operates in a brute-force way as it only includes information about how to traverse the tree and how to identify leaf and goal nodes. Uninformed search applies a way in which search tree is searched without any information about the search space like initial state operators and test for the goal, so it is also called blind search.It examines each node of the tree until it achieves the goal node.

**It can be divided into five main types:**

* Breadth-first search
* Uniform cost search
* Depth-first search
* Iterative deepening depth-first search
* Bidirectional Search

### **Informed Search**

Informed search algorithms use domain knowledge. In an informed search, problem information is available which can guide the search. Informed search strategies can find a solution more efficiently than an uninformed search strategy. Informed search is also called a Heuristic search.

A heuristic is a way which might not always be guaranteed for best solutions but guaranteed to find a good solution in reasonable time.

Informed search can solve much complex problem which could not be solved in another way.

An example of informed search algorithms is a traveling salesman problem.

1. Greedy Search
2. A\* Search

# 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.**

**Following are the various types of uninformed search algorithms:**

1. **Breadth-first Search**
2. **Depth-first Search**
3. **Depth-limited Search**
4. **Iterative deepening depth-first search**
5. **Uniform cost search**
6. **Bidirectional Search**

## 1. Breadth-first Search:

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

**Advantages:**

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

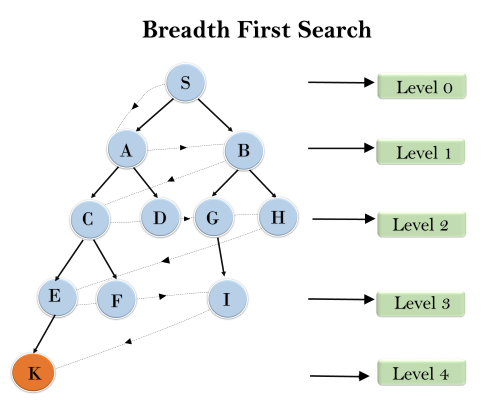
**Disadvantages:**

* 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+b2+b3+.......+ bd= O (bd)**

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

**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.

#### **Note: 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:**

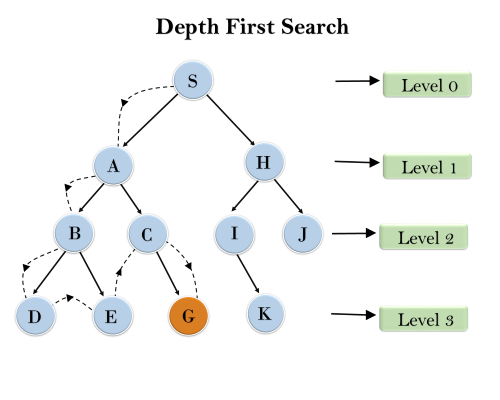
* There is the possibility that many states keep re-occurring, and there is no guarantee of finding the solution.
* DFS algorithm goes for deep down searching and sometime it may go to the infinite loop.

### **Example:**

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

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

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



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

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

**T(n)= 1+ n2+ n3 +.........+ nm=O(nm)**

**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(bℓ)**.

**Space Complexity:** Space complexity of DLS algorithm is O**(b×ℓ)**.

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

## 4. Uniform-cost Search Algorithm:

Uniform-cost search is a searching algorithm used for traversing a weighted tree or graph. This algorithm comes into play when a different cost is available for each edge. The primary goal of the uniform-cost search is to find a path to the goal node which has the lowest cumulative cost. Uniform-cost search expands nodes according to their path costs 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.

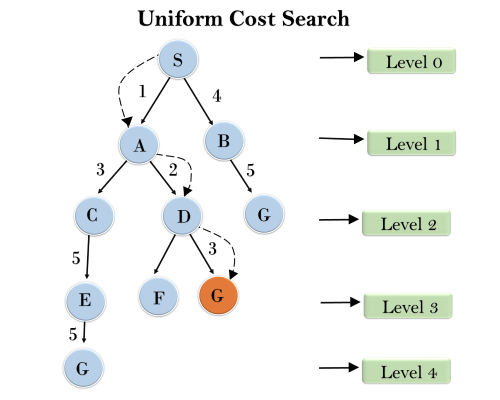
**Advantages:**

* 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**O(b1 + [C\*/ε])/**.

**Space Complexity:**

The same logic is for space complexity so, the worst-case space complexity of Uniform-cost search is **O(b1 + [C\*/ε])**.

**Optimal:**

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

## 5. Iterative deepeningdepth-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:**

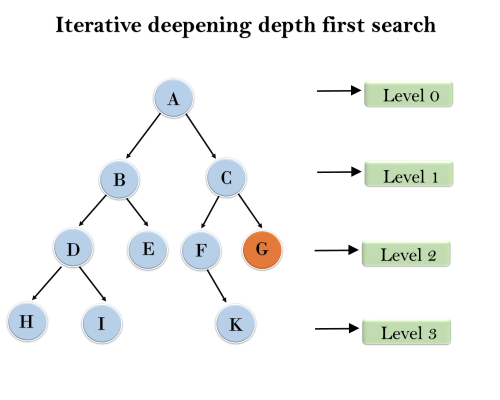
* Itcombines the benefits of BFS and DFS search algorithm in terms of fast search and memory efficiency.

**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  
4'th Iteration------>A, B, D, H, I, E, C, F, K, G  
In the fourth iteration, the algorithm will find the goal node.

**Completeness:**

This algorithm is complete is 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 **O(bd)**.

**Space Complexity:**

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

**Optimal:**

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

## 6. Bidirectional Search Algorithm:

**Bidirectional search algorithm runs two simultaneous searches, one form initial state called as forward-search and other from goal node called as backward-search, to find the goal node. Bidirectional search replaces one single search graph with two small subgraphs in which one starts the search from an initial vertex and other starts from goal vertex. The search stops when these two graphs intersect each other.**

**Bidirectional search can use search techniques such as BFS, DFS, DLS, etc.**

**Advantages:**

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

**Disadvantages:**

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

### **Example:**

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

The algorithm terminates at node 9 where two searches meet.

# 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.

In the informed search we will discuss two main algorithms which are given below:

* **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:**

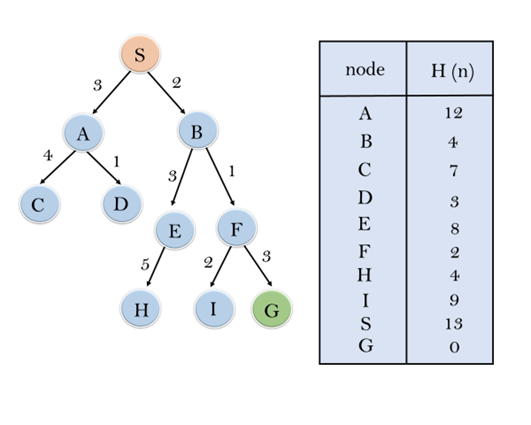
* Best first search can switch between BFS and DFS by gaining the advantages of both the algorithms.
* This algorithm is more efficient than BFS and DFS algorithms.

### **Disadvantages:**

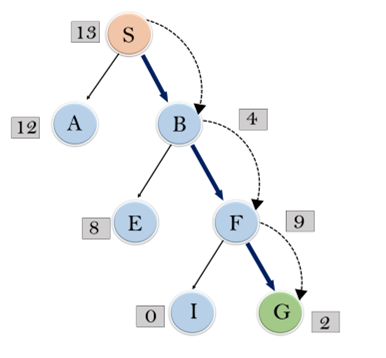
* It can behave as an unguided depth-first search in the worst case scenario.
* It can get stuck in a loop as DFS.
* This algorithm is not optimal.

### **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.



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(bm).

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

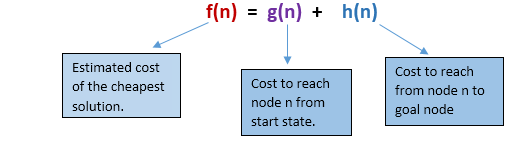
**Complete:** Greedy best-first search is also incomplete, even if the given state space is finite.

**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 node is 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.

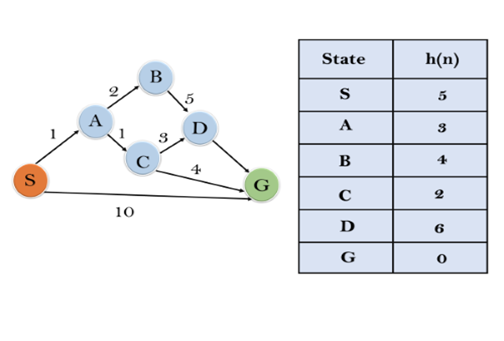
### **Disadvantages:**

* 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.

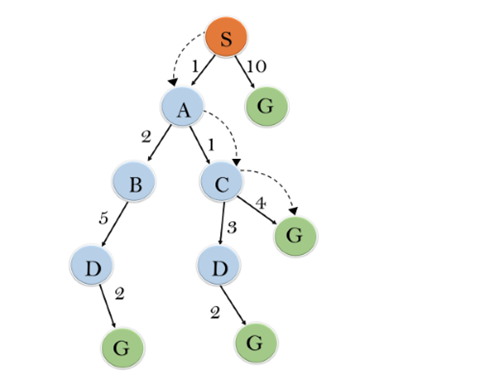
### **Example:**

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.



**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.

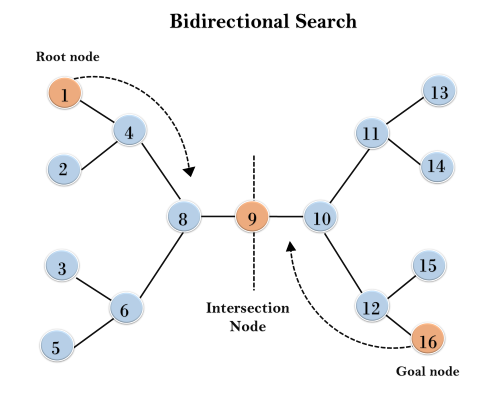
**Optimal:** A\* search algorithm is optimal if it follows below two conditions:

* **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)**



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

**Time Complexity:** Time complexity of bidirectional search using BFS is **O(bd)**.

**Space Complexity:** Space complexity of bidirectional search is **O(bd)**.

**Optimal:** Bidirectional search is Optimal.

# 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:

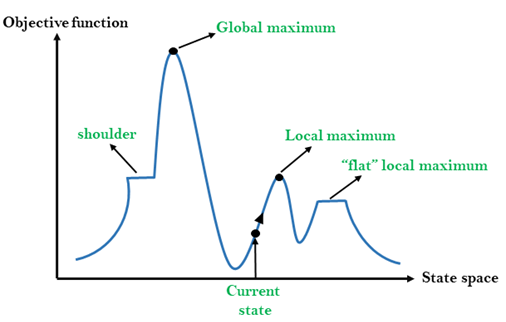
* **Generate and Test variant:** Hill Climbing is the variant of Generate and Test method. The Generate and Test method produce feedback which helps to decide which direction to move in the search space.
* **Greedy approach:** Hill-climbing algorithm search moves in the direction which optimizes the 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.

ADVERTISING



## 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

### **Algorithm for Steepest-Ascent hill climbing:**

* **Step 1:** Evaluate the initial state, if it is goal state then return success and stop, else make current state as initial state.
* **Step 2:** Loop until a solution is found or the current state does not change.
  1. Let SUCC be a state such that any successor of the current state will be better than it.
  2. For each operator that applies to the current state:
     1. Apply the new operator and generate a new state.
     2. Evaluate the new state.
     3. If it is goal state, then return it and quit, else compare it to the SUCC.
     4. If it is better than SUCC, then set new state as SUCC.
     5. If the SUCC is better than the current state, then set current state to SUCC.
* **Step 5:** Exit.

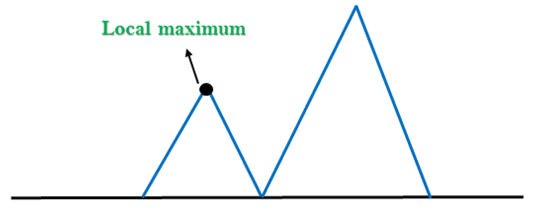
### **3. Stochastic hill climbing:**

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.

## Problems in Hill Climbing Algorithm:

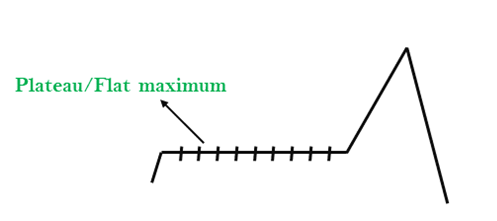
**1. Local Maximum:** A local maximum is a peak state in the landscape which is better than each of its neighboring states, but there is another state also present which is higher than the local maximum.

**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.



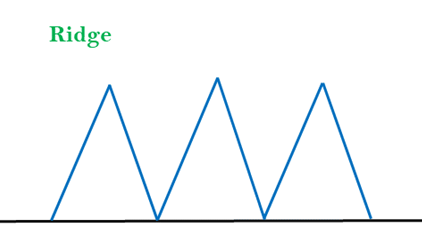
**2. Plateau:** A plateau is the flat area of the search space in which all the neighbor states of the current state contains the same value, because of this algorithm does not find any best direction to move. A hill-climbing search might be lost in the plateau area.

**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:** A ridge is a special form of the local maximum. It has an area which is higher than its surrounding areas, but itself has a slope, and cannot be reached in a single move.

**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.

# 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.

1. First, evaluate the difference between Initial State and final State.
2. Select the various operators which can be applied for each difference.
3. 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:

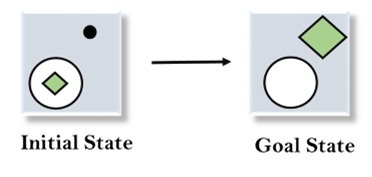
Let's we take Current state as CURRENT and Goal State as GOAL, then following are the steps for the MEA algorithm.

* **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 O?s 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:**

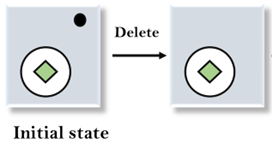
To solve the above problem, we will first find the differences between initial states and goal states, and for each difference, we will generate a new state and will apply the operators. The operators we have for this problem are:

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

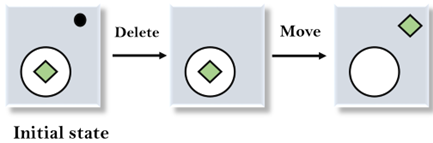
**1. Evaluating the initial state:** In the first step, we will evaluate the initial state and will compare the initial and Goal state to find the differences between both states.



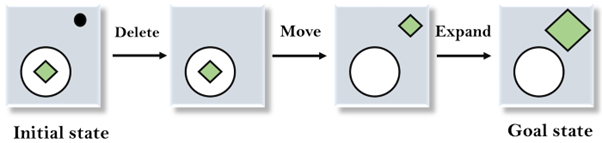
**2. Applying Delete operator:** As we can check the first difference is that in goal state there is no dot symbol which is present in the initial state, so, first we will apply the **Delete operator** to remove this dot.



**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.



# Adversarial Search

**Adversarial search is a search, where we examine the problem which arises when we try to plan ahead of the world and other agents are planning against us.**

* In previous topics, we have studied the search strategies which are only associated with a single agent that aims to find the solution which often expressed in the form of a sequence of actions.
* But, there might be some situations where more than one agent is searching for the solution in the same search space, and this situation usually occurs in game playing.
* The environment with more than one agent is termed as **multi-agent environment**, in which each agent is an opponent of other agent and playing against each other. Each agent needs to consider the action of other agent and effect of that action on their performance.
* So, **Searches in which two or more players with conflicting goals are trying to explore the same search space for the solution, are called adversarial searches, often known as Games**.
* Games are modeled as a Search problem and heuristic evaluation function, and these are the two main factors which help to model and solve games in AI.

## Types of Games in AI:

|  |  |  |
| --- | --- | --- |
|  | **Deterministic** | **Chance Moves** |
| **Perfect information** | Chess, Checkers, go, Othello | Backgammon, monopoly |
| **Imperfect information** | Battleships, blind, tic-tac-toe | Bridge, poker, scrabble, nuclear war |

* **Perfect information:** A game with the perfect information is that in which agents can look into the complete board. Agents have all the information about the game, and they can see each other moves also. Examples are Chess, Checkers, Go, etc.
* **Imperfect information:** If in a game agents do not have all information about the game and not aware with what's going on, such type of games are called the game with imperfect information, such as tic-tac-toe, Battleship, blind, Bridge, etc.
* **Deterministic games:** Deterministic games are those games which follow a strict pattern and set of rules for the games, and there is no randomness associated with them. Examples are chess, Checkers, Go, tic-tac-toe, etc.
* **Non-deterministic games:** Non-deterministic are those games which have various unpredictable events and has a factor of chance or luck. This factor of chance or luck is introduced by either dice or cards. These are random, and each action response is not fixed. Such games are also called as stochastic games.  
  Example: Backgammon, Monopoly, Poker, etc.

#### **Note: In this topic, we will discuss deterministic games, fully observable environment, zero-sum, and where each agent acts alternatively.**

## Zero-Sum Game

* Zero-sum games are adversarial search which involves pure competition.
* In Zero-sum game each agent's gain or loss of utility is exactly balanced by the losses or gains of utility of another agent.
* One player of the game try to maximize one single value, while other player tries to minimize it.
* Each move by one player in the game is called as ply.
* Chess and tic-tac-toe are examples of a Zero-sum game.

### **Zero-sum game: Embedded thinking**

The Zero-sum game involved embedded thinking in which one agent or player is trying to figure out:

* What to do.
* How to decide the move
* Needs to think about his opponent as well
* The opponent also thinks what to do

Each of the players is trying to find out the response of his opponent to their actions. This requires embedded thinking or backward reasoning to solve the game problems in AI.

### **Formalization of the problem:**

**A game can be defined as a type of search in AI which can be formalized of the following elements:**

* **Initial state:** It specifies how the game is set up at the start.
* **Player(s):** It specifies which player has moved in the state space.
* **Action(s):** It returns the set of legal moves in state space.
* **Result(s, a):** It is the transition model, which specifies the result of moves in the state space.
* **Terminal-Test(s):** Terminal test is true if the game is over, else it is false at any case. The state where the game ends is called terminal states.
* **Utility(s, p):** A utility function gives the final numeric value for a game that ends in terminal states s for player p. It is also called payoff function. For Chess, the outcomes are a win, loss, or draw and its payoff values are +1, 0, ½. And for tic-tac-toe, utility values are +1, -1, and 0.

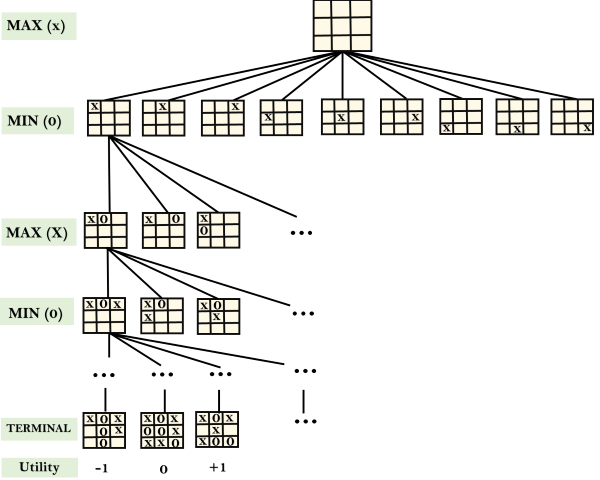
## Game tree:

A game tree is a tree where nodes of the tree are the game states and Edges of the tree are the moves by players. Game tree involves initial state, actions function, and result Function.

**Example: Tic-Tac-Toe game tree:**

The following figure is showing part of the game-tree for tic-tac-toe game. Following are some key points of the game:

* There are two players MAX and MIN.
* Players have an alternate turn and start with MAX.
* MAX maximizes the result of the game tree
* MIN minimizes the result.



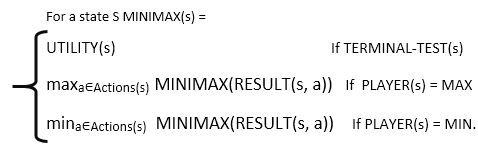
**Example Explanation:**

* From the initial state, MAX has 9 possible moves as he starts first. MAX place x and MIN place o, and both player plays alternatively until we reach a leaf node where one player has three in a row or all squares are filled.
* Both players will compute each node, minimax, the minimax value which is the best achievable utility against an optimal adversary.
* Suppose both the players are well aware of the tic-tac-toe and playing the best play. Each player is doing his best to prevent another one from winning. MIN is acting against Max in the game.
* So in the game tree, we have a layer of Max, a layer of MIN, and each layer is called as **Ply**. Max place x, then MIN puts o to prevent Max from winning, and this game continues until the terminal node.
* In this either MIN wins, MAX wins, or it's a draw. This game-tree is the whole search space of possibilities that MIN and MAX are playing tic-tac-toe and taking turns alternately.

Hence adversarial Search for the minimax procedure works as follows:

* It aims to find the optimal strategy for MAX to win the game.
* It follows the approach of Depth-first search.
* In the game tree, optimal leaf node could appear at any depth of the tree.
* Propagate the minimax values up to the tree until the terminal node discovered.

In a given game tree, the optimal strategy can be determined from the minimax value of each node, which can be written as MINIMAX(n). MAX prefer to move to a state of maximum value and MIN prefer to move to a state of minimum value then:



Mini-Max Algorithm in Artificial Intelligence

* Mini-max algorithm is a recursive or backtracking algorithm which is used in decision-making and game theory. It provides an optimal move for the player assuming that opponent is also playing optimally.
* Mini-Max algorithm uses recursion to search through the game-tree.
* Min-Max algorithm is mostly used for game playing in AI. Such as Chess, Checkers, tic-tac-toe, go, and various tow-players game. This Algorithm computes the minimax decision for the current state.
* In this algorithm two players play the game, one is called MAX and other is called MIN.
* Both the players fight it as the opponent player gets the minimum benefit while they get the maximum benefit.
* Both Players of the game are opponent of each other, where MAX will select the maximized value and MIN will select the minimized value.
* The minimax algorithm performs a depth-first search algorithm for the exploration of the complete game tree.
* The minimax algorithm proceeds all the way down to the terminal node of the tree, then backtrack the tree as the recursion.

Pseudo-code for MinMax Algorithm:

1. function minimax(node, depth, maximizingPlayer) is
2. **if** depth ==0 or node is a terminal node then
3. **return** **static** evaluation of node
5. **if** MaximizingPlayer then      // for Maximizer Player
6. maxEva= -infinity
7. **for** each child of node **do**
8. eva= minimax(child, depth-1, **false**)
9. maxEva= max(maxEva,eva)        //gives Maximum of the values
10. **return** maxEva
12. **else**                         // for Minimizer player
13. minEva= +infinity
14. **for** each child of node **do**
15. eva= minimax(child, depth-1, **true**)
16. minEva= min(minEva, eva)         //gives minimum of the values
17. **return** minEva

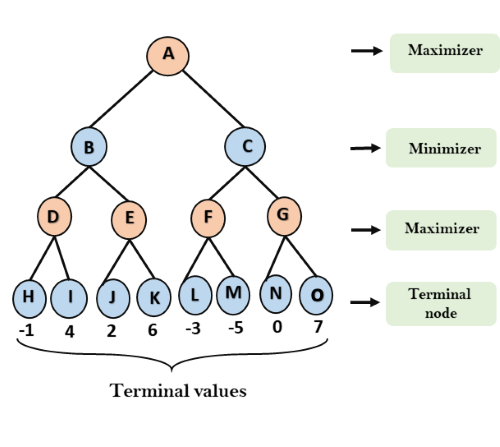
**Initial call:**

**Minimax(node, 3, true)**

Working of Min-Max Algorithm:

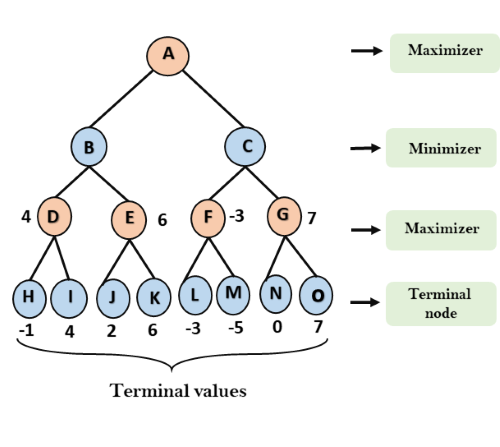
* The working of the minimax algorithm can be easily described using an example. Below we have taken an example of game-tree which is representing the two-player game.
* In this example, there are two players one is called Maximizer and other is called Minimizer.
* Maximizer will try to get the Maximum possible score, and Minimizer will try to get the minimum possible score.
* This algorithm applies DFS, so in this game-tree, we have to go all the way through the leaves to reach the terminal nodes.
* At the terminal node, the terminal values are given so we will compare those value and backtrack the tree until the initial state occurs. Following are the main steps involved in solving the two-player game tree:

**Step-1:** In the first step, the algorithm generates the entire game-tree and apply the utility function to get the utility values for the terminal states. In the below tree diagram, let's take A is the initial state of the tree. Suppose maximizer takes first turn which has worst-case initial value =- infinity, and minimizer will take next turn which has worst-case initial value = +infinity.



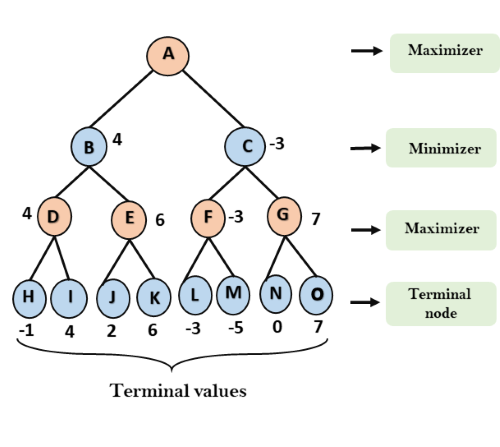
**Step 2:** Now, first we find the utilities value for the Maximizer, its initial value is -∞, so we will compare each value in terminal state with initial value of Maximizer and determines the higher nodes values. It will find the maximum among the all.

* For node D         max(-1,- -∞) => max(-1,4)= 4
* For Node E         max(2, -∞) => max(2, 6)= 6
* For Node F         max(-3, -∞) => max(-3,-5) = -3
* For node G         max(0, -∞) = max(0, 7) = 7



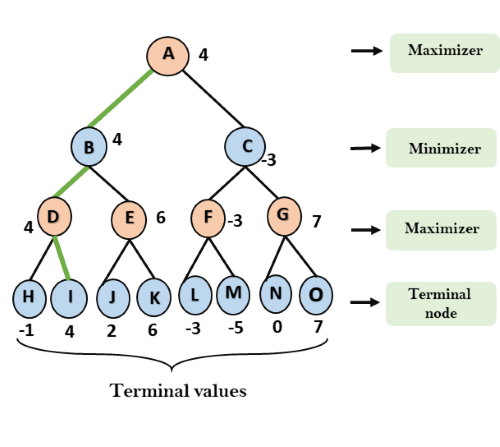
**Step 3:** In the next step, it's a turn for minimizer, so it will compare all nodes value with +∞, and will find the 3rd layer node values.

* For node B= min(4,6) = 4
* For node C= min (-3, 7) = -3



**Step 3:** Now it's a turn for Maximizer, and it will again choose the maximum of all nodes value and find the maximum value for the root node. In this game tree, there are only 4 layers, hence we reach immediately to the root node, but in real games, there will be more than 4 layers.

* For node A max(4, -3)= 4



That was the complete workflow of the minimax two player game.

Properties of Mini-Max algorithm:

* **Complete-** Min-Max algorithm is Complete. It will definitely find a solution (if exist), in the finite search tree.
* **Optimal-** Min-Max algorithm is optimal if both opponents are playing optimally.
* **Time complexity-** As it performs DFS for the game-tree, so the time complexity of Min-Max algorithm is **O(bm)**, where b is branching factor of the game-tree, and m is the maximum depth of the tree.
* **Space Complexity-** Space complexity of Mini-max algorithm is also similar to DFS which is **O(bm)**.

Limitation of the minimax Algorithm:

The main drawback of the minimax algorithm is that it gets really slow for complex games such as Chess, go, etc. This type of games has a huge branching factor, and the player has lots of choices to decide. This limitation of the minimax algorithm can be improved from **alpha-beta pruning** which we have discussed in the next topic.

# Alpha-Beta Pruning

* Alpha-beta pruning is a modified version of the minimax algorithm. It is an optimization technique for the minimax algorithm.
* As we have seen in the minimax search algorithm that the number of game states it has to examine are exponential in depth of the tree. Since we cannot eliminate the exponent, but we can cut it to half. Hence there is a technique by which without checking each node of the game tree we can compute the correct minimax decision, and this technique is called **pruning**. This involves two threshold parameter Alpha and beta for future expansion, so it is called **alpha-beta pruning**. It is also called as **Alpha-Beta Algorithm**.
* Alpha-beta pruning can be applied at any depth of a tree, and sometimes it not only prune the tree leaves but also entire sub-tree.
* The two-parameter can be defined as:
  1. **Alpha:** The best (highest-value) choice we have found so far at any point along the path of Maximizer. The initial value of alpha is **-∞**.
  2. **Beta:** The best (lowest-value) choice we have found so far at any point along the path of Minimizer. The initial value of beta is **+∞**.
* The Alpha-beta pruning to a standard minimax algorithm returns the same move as the standard algorithm does, but it removes all the nodes which are not really affecting the final decision but making algorithm slow. Hence by pruning these nodes, it makes the algorithm fast.

#### **Note: To better understand this topic, kindly study the minimax algorithm.**

## Condition for Alpha-beta pruning:

The main condition which required for alpha-beta pruning is:

1. α>=β

## Key points about alpha-beta pruning:

* The Max player will only update the value of alpha.
* The Min player will only update the value of beta.
* While backtracking the tree, the node values will be passed to upper nodes instead of values of alpha and beta.
* We will only pass the alpha, beta values to the child nodes.

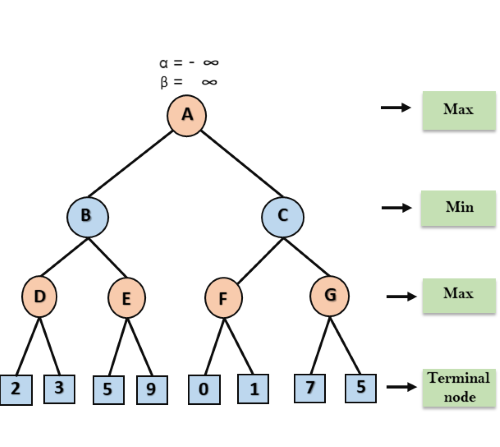
## Pseudo-code for Alpha-beta Pruning:

1. function minimax(node, depth, alpha, beta, maximizingPlayer) is
2. **if** depth ==0 or node is a terminal node then
3. **return** **static** evaluation of node
5. **if** MaximizingPlayer then      // for Maximizer Player
6. maxEva= -infinity
7. **for** each child of node **do**
8. eva= minimax(child, depth-1, alpha, beta, False)
9. maxEva= max(maxEva, eva)
10. alpha= max(alpha, maxEva)
11. **if** beta<=alpha
12. **break**
13. **return** maxEva
15. **else**                         // for Minimizer player
16. minEva= +infinity
17. **for** each child of node **do**
18. eva= minimax(child, depth-1, alpha, beta, **true**)
19. minEva= min(minEva, eva)
20. beta= min(beta, eva)
21. **if** beta<=alpha
22. **break**
23. **return** minEva

## Working of Alpha-Beta Pruning:

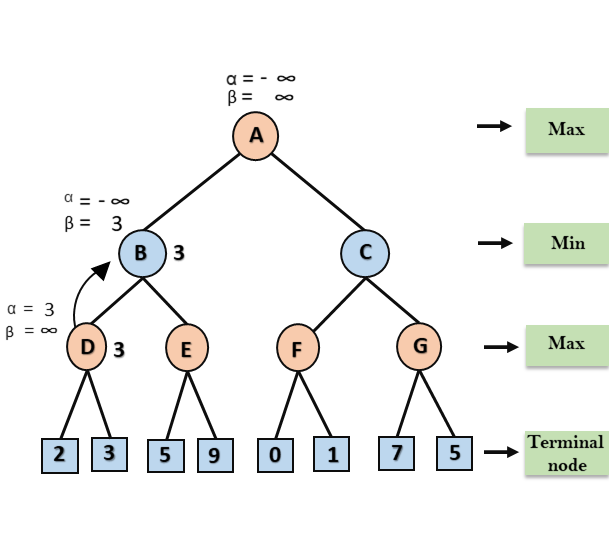
Let's take an example of two-player search tree to understand the working of Alpha-beta pruning

**Step 1:** At the first step the, Max player will start first move from node A where α= -∞ and β= +∞, these value of alpha and beta passed down to node B where again α= -∞ and β= +∞, and Node B passes the same value to its child D.



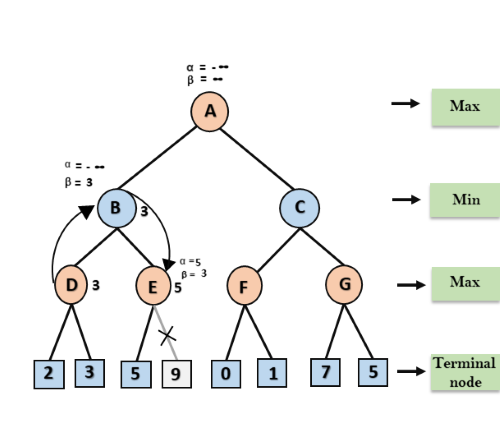
**Step 2:** At Node D, the value of α will be calculated as its turn for Max. The value of α is compared with firstly 2 and then 3, and the max (2, 3) = 3 will be the value of α at node D and node value will also 3.

**Step 3:** Now algorithm backtrack to node B, where the value of β will change as this is a turn of Min, Now β= +∞, will compare with the available subsequent nodes value, i.e. min (∞, 3) = 3, hence at node B now α= -∞, and β= 3.



In the next step, algorithm traverse the next successor of Node B which is node E, and the values of α= -∞, and β= 3 will also be passed.

**Step 4:** At node E, Max will take its turn, and the value of alpha will change. The current value of alpha will be compared with 5, so max (-∞, 5) = 5, hence at node E α= 5 and β= 3, where α>=β, so the right successor of E will be pruned, and algorithm will not traverse it, and the value at node E will be 5.



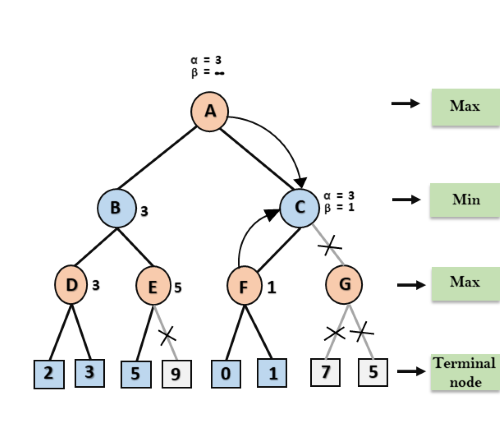
**Step 5:** At next step, algorithm again backtrack the tree, from node B to node A. At node A, the value of alpha will be changed the maximum available value is 3 as max (-∞, 3)= 3, and β= +∞, these two values now passes to right successor of A which is Node C.

At node C, α=3 and β= +∞, and the same values will be passed on to node F.

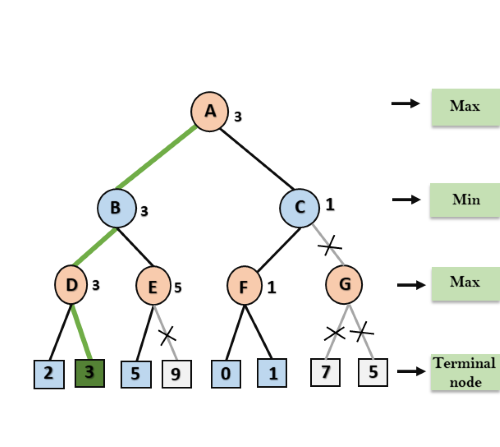
**Step 6:** At node F, again the value of α will be compared with left child which is 0, and max(3,0)= 3, and then compared with right child which is 1, and max(3,1)= 3 still α remains 3, but the node value of F will become 1.



**Step 7:** Node F returns the node value 1 to node C, at C α= 3 and β= +∞, here the value of beta will be changed, it will compare with 1 so min (∞, 1) = 1. Now at C, α=3 and β= 1, and again it satisfies the condition α>=β, so the next child of C which is G will be pruned, and the algorithm will not compute the entire sub-tree G.



**Step 8:** C now returns the value of 1 to A here the best value for A is max (3, 1) = 3. Following is the final game tree which is the showing the nodes which are computed and nodes which has never computed. Hence the optimal value for the maximizer is 3 for this example.



## Move Ordering in Alpha-Beta pruning:

The effectiveness of alpha-beta pruning is highly dependent on the order in which each node is examined. Move order is an important aspect of alpha-beta pruning.

It can be of two types:

* **Worst ordering:** In some cases, alpha-beta pruning algorithm does not prune any of the leaves of the tree, and works exactly as minimax algorithm. In this case, it also consumes more time because of alpha-beta factors, such a move of pruning is called worst ordering. In this case, the best move occurs on the right side of the tree. The time complexity for such an order is O(bm).
* **Ideal ordering:** The ideal ordering for alpha-beta pruning occurs when lots of pruning happens in the tree, and best moves occur at the left side of the tree. We apply DFS hence it first search left of the tree and go deep twice as minimax algorithm in the same amount of time. Complexity in ideal ordering is O(bm/2).

## Rules to find good ordering:

Following are some rules to find good ordering in alpha-beta pruning:

* Occur the best move from the shallowest node.
* Order the nodes in the tree such that the best nodes are checked first.
* Use domain knowledge while finding the best move. Ex: for Chess, try order: captures first, then threats, then forward moves, backward moves.
* We can bookkeep the states, as there is a possibility that states may repeat.