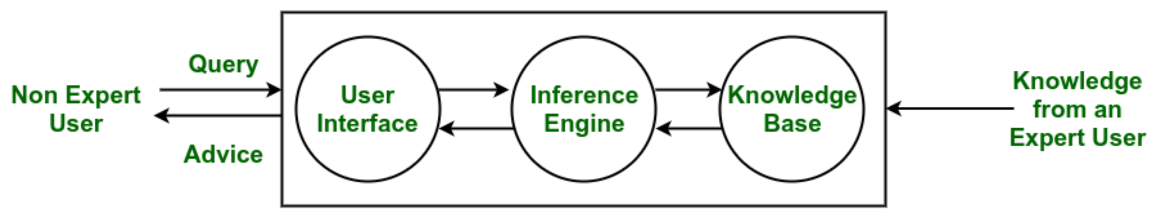
There are many examples of an expert system. Some of them are given below –

- **MYCIN –**
  One of the earliest expert systems based on backward chaining. It can identify various bacteria that can cause severe infections and can also recommend drugs based on the person's weight.
- **DENDRAL –**
  It was an artificial intelligence-based expert system used for chemical analysis. It used a substance's spectrographic data to predict its molecular structure.
- **R1/XCON –**
  It could select specific software to generate a computer system wished by the user.
- **PXDES –**
  It could easily determine the type and the degree of lung cancer in a patient based on the data.
- **CaDet –**
  It is a clinical support system that could identify cancer in its early stages in patients.
- **DXplain –**
  It was also a clinical support system that could suggest a variety of diseases based on the findings of the doctor.

Components of an Expert System :



Architecture of an Expert System

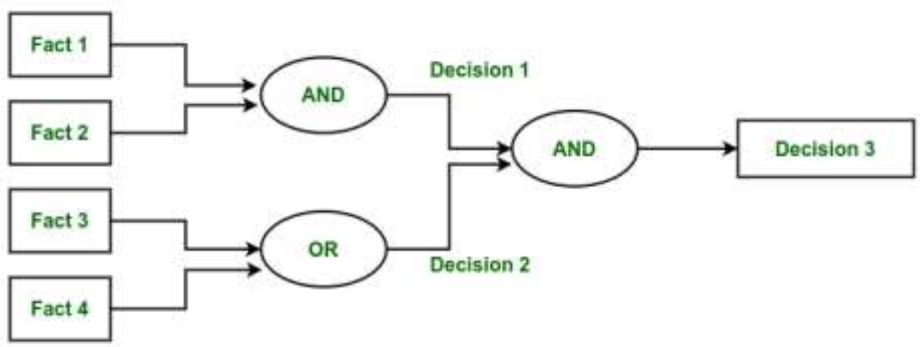
- **Knowledge Base –**
  The knowledge base represents facts and rules. It consists of knowledge in a particular domain as well as rules to solve a problem, procedures and intrinsic data relevant to the domain.
- **Inference Engine –**
  The function of the inference engine is to fetch the relevant knowledge from the knowledge base, interpret it and to find a solution relevant to the user's problem. The inference engine acquires the rules from its knowledge base and applies them to the known facts to infer new facts. Inference engines can also include an explanation and debugging abilities.
- **Knowledge Acquisition and Learning Module –**
  The function of this component is to allow the expert system to acquire more and more knowledge from various sources and store it in the knowledge base.
- **User Interface –**
  This module makes it possible for a non-expert user to interact with the expert system and find a solution to the problem.

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

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

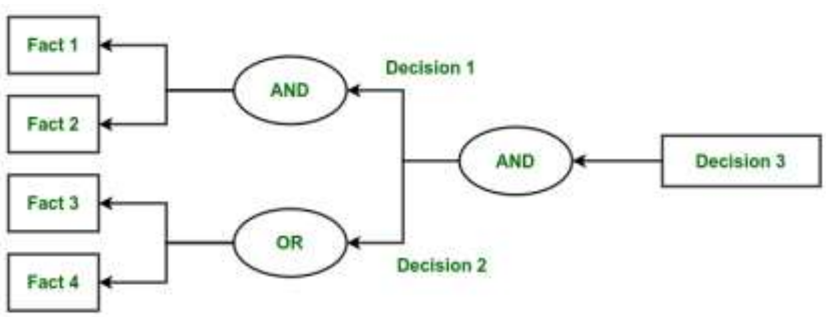
- Forward Chaining
- Backward Chaining

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



Forward Chaining

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



Backward Chaining

Characteristics of an Expert System :

- Human experts are perishable, but an expert system is permanent.
- It helps to distribute the expertise of a human.
- One expert system may contain knowledge from more than one human experts thus making the solutions more efficient.
- It decreases the cost of consulting an expert for various domains such as medical diagnosis.
- They use a knowledge base and inference engine.

- Expert systems can solve complex problems by deducing new facts through existing facts of knowledge, represented mostly as if-then rules rather than through conventional procedural code.
- Expert systems were among the first truly successful forms of artificial intelligence (AI) software.

Limitations :

- Do not have human-like decision-making power.
- Cannot possess human capabilities.
- Cannot produce correct result from less amount of knowledge.
- Requires excessive training.

Advantages :

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

Disadvantages:

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

Applications :

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

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

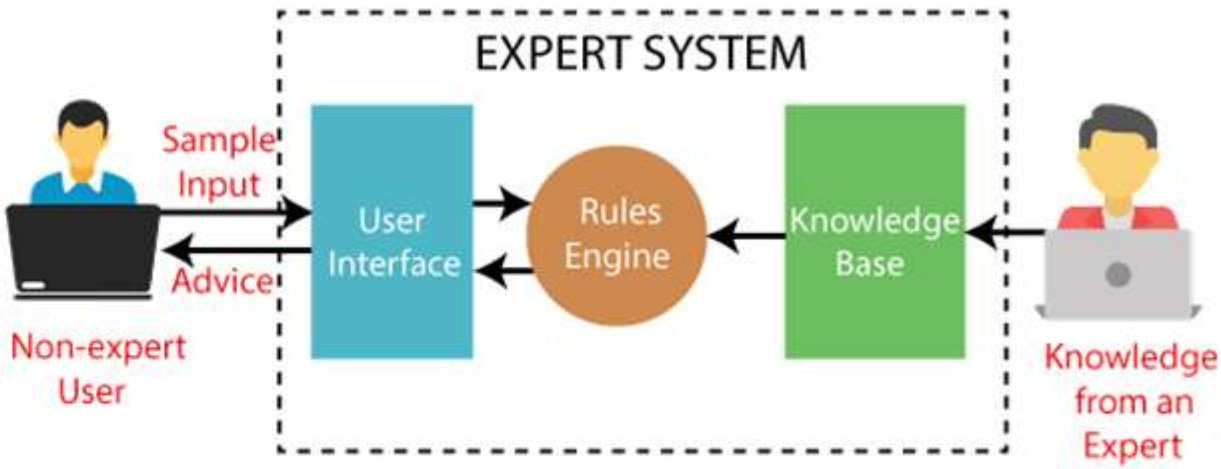
What is an Expert System?

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

The expert system is a part of AI, and the first ES was developed in the year 1970, which was the first successful approach of artificial intelligence. It solves the most complex issue as an expert by extracting the knowledge stored in its knowledge base. The system helps in decision making for compsex problems using **both facts and heuristics like a human expert**. It is called so because it contains the expert knowledge of a specific domain and can solve any complex problem of that particular domain. These systems are designed for a specific domain, such as **medicine, science,** etc.

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

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



Note: It is important to remember that an expert system is not used to replace the human experts; instead, it is used to assist the human in making a complex decision. These systems do not have human capabilities of thinking and work on the basis of the knowledge base of the particular domain.

Below are some popular examples of the Expert System:

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

- **CaDeT:** The CaDet expert system is a diagnostic support system that can detect cancer at early stages.

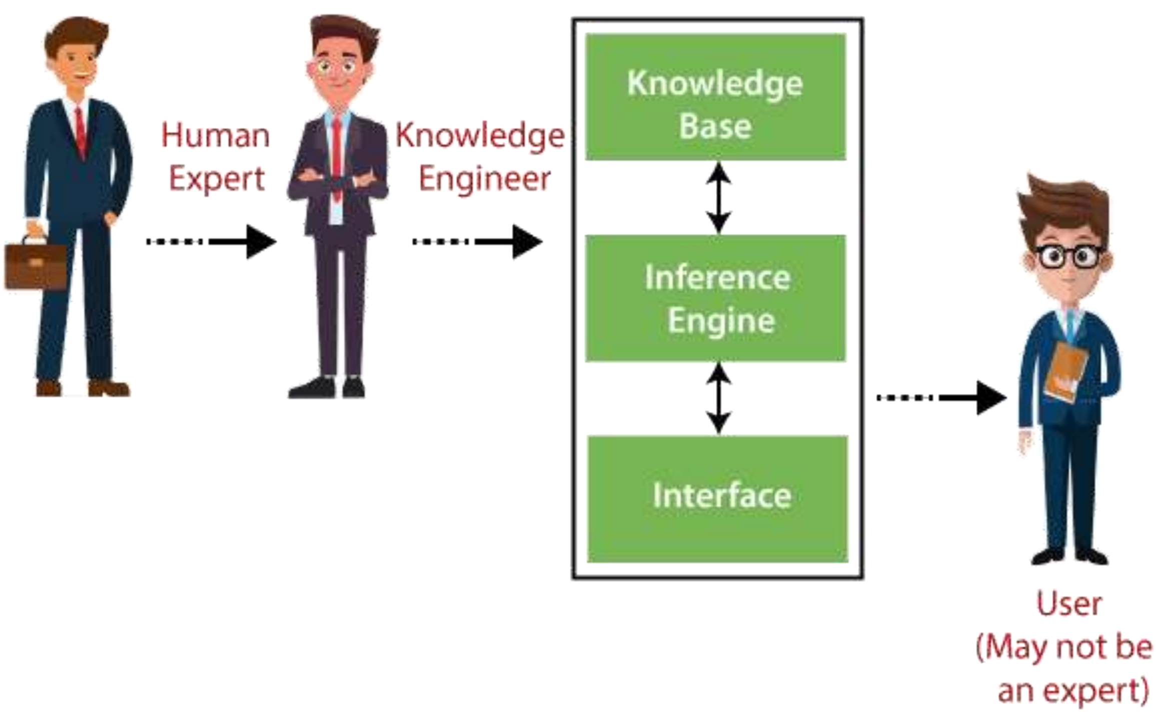
Characteristics of Expert System

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

Components of Expert System

An expert system mainly consists of three components:

- **User Interface**
- **Inference Engine**
- **Knowledge Base**



1. User Interface

With the help of a user interface, the expert system interacts with the user, takes queries as an input in a readable format, and passes it to the inference engine. After getting the response from the inference engine, it displays the output to the user. In other words, **it is an interface that helps a non-expert user to communicate with the expert system to find a solution**.

2. Inference Engine(Rules of Engine)

- The inference engine is known as the brain of the expert system as it is the main processing unit of the system. It applies inference rules to the knowledge base to derive a conclusion or deduce new information. It helps in deriving an error-free solution of queries asked by the user.
- With the help of an inference engine, the system extracts the knowledge from the knowledge base.
- There are two types of inference engine:
- **Deterministic Inference engine:** The conclusions drawn from this type of inference engine are assumed to be true. It is based on **facts** and **rules**.
- **Probabilistic Inference engine:** This type of inference engine contains uncertainty in conclusions, and based on the probability.

Inference engine uses the below modes to derive the solutions:

- **Forward Chaining:** It starts from the known facts and rules, and applies the inference rules to add their conclusion to the known facts.
- **Backward Chaining:** It is a backward reasoning method that starts from the goal and works backward to prove the known facts.

3. Knowledge Base

- The knowledgebase is a type of storage that stores knowledge acquired from the different experts of the particular domain. It is considered as big storage of knowledge. The more the knowledge base, the more precise will be the Expert System.
- It is similar to a database that contains information and rules of a particular domain or subject.
- One can also view the knowledge base as collections of objects and their attributes. Such as a Lion is an object and its attributes are it is a mammal, it is not a domestic animal, etc.

Components of Knowledge Base

- **Factual Knowledge:** The knowledge which is based on facts and accepted by knowledge engineers comes under factual knowledge.
- **Heuristic Knowledge:** This knowledge is based on practice, the ability to guess, evaluation, and experiences.

Knowledge Representation: It is used to formalize the knowledge stored in the knowledge base using the If-else rules.

Knowledge Acquisitions: It is the process of extracting, organizing, and structuring the domain knowledge, specifying the rules to acquire the knowledge from various experts, and store that knowledge into the knowledge base.

Development of Expert System

Here, we will explain the working of an expert system by taking an example of MYCIN ES. Below are some steps to build an MYCIN:

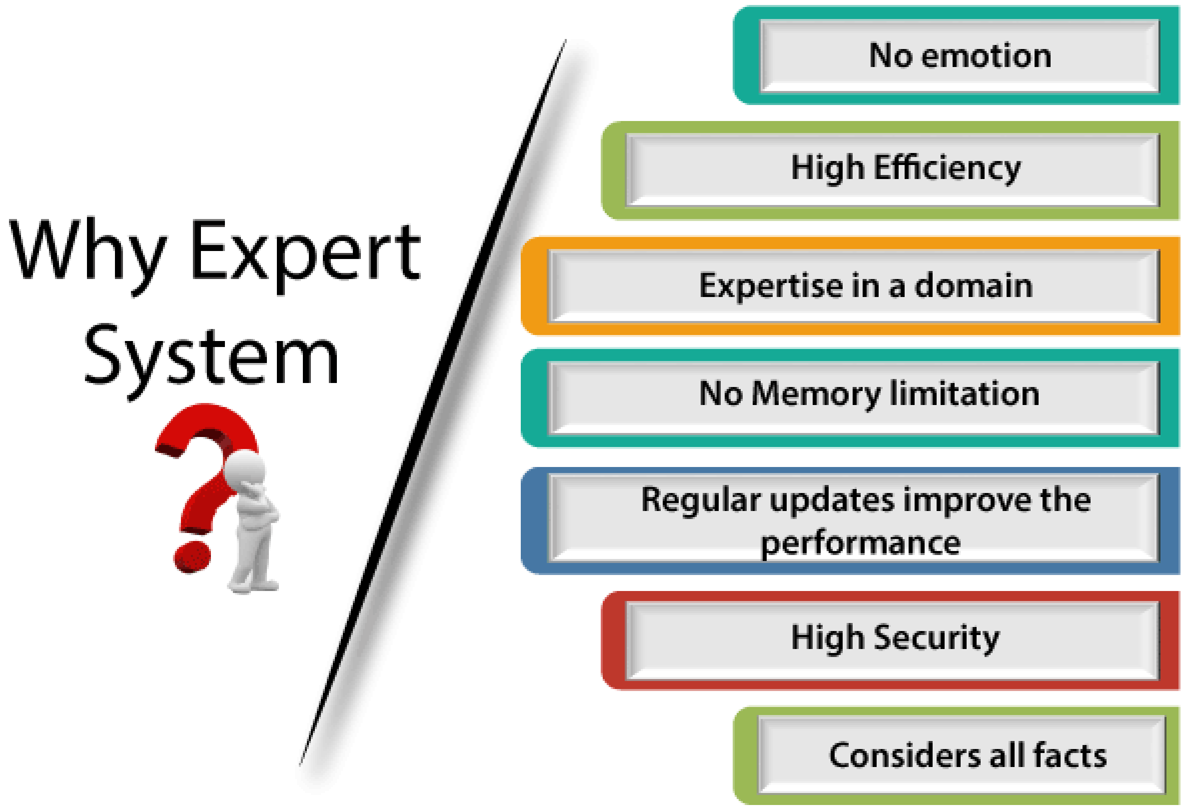
- Firstly, ES should be fed with expert knowledge. In the case of MYCIN, human experts specialized in the medical field of bacterial infection, provide information about the causes, symptoms, and other knowledge in that domain.
- The KB of the MYCIN is updated successfully. In order to test it, the doctor provides a new problem to it. The problem is to identify the presence of the bacteria by inputting the details of a patient, including the symptoms, current condition, and medical history.
- The ES will need a questionnaire to be filled by the patient to know the general information about the patient, such as gender, age, etc.
- Now the system has collected all the information, so it will find the solution for the problem by applying if-then rules using the inference engine and using the facts stored within the KB.
- In the end, it will provide a response to the patient by using the user interface.

Participants in the development of Expert System

There are three primary participants in the building of Expert System:

1. **Expert:** The success of an ES much depends on the knowledge provided by human experts. These experts are those persons who are specialized in that specific domain.
2. **Knowledge Engineer:** Knowledge engineer is the person who gathers the knowledge from the domain experts and then codifies that knowledge to the system according to the formalism.
3. **End-User:** This is a particular person or a group of people who may not be experts, and working on the expert system needs the solution or advice for his queries, which are complex.

Why Expert System?



Before using any technology, we must have an idea about why to use that technology and hence the same for the ES. Although we have human experts in every field, then what is the need to develop a computer-based system. So below are the points that are describing the need of the ES:

1. **No memory Limitations:** It can store as much data as required and can memorize it at the time of its application. But for human experts, there are some limitations to memorize all things at every time.
2. **High Efficiency:** If the knowledge base is updated with the correct knowledge, then it provides a highly efficient output, which may not be possible for a human.
3. **Expertise in a domain:** There are lots of human experts in each domain, and they all have different skills, different experiences, and different skills, so it is not easy to get a final output for the query. But if we put the knowledge gained from human experts into the expert system, then it provides an efficient output by mixing all the facts and knowledge
4. **Not affected by emotions:** These systems are not affected by human emotions such as fatigue, anger, depression, anxiety, etc.. Hence the performance remains constant.
5. **High security:** These systems provide high security to resolve any query.
6. **Considers all the facts:** To respond to any query, it checks and considers all the available facts and provides the result accordingly. But it is possible that a human expert may not consider some facts due to any reason.

7. **Regular updates improve the performance:** If there is an issue in the result provided by the expert systems, we can improve the performance of the system by updating the knowledge base.

Capabilities of the Expert System

Below are some capabilities of an Expert System:

- **Advising:** It is capable of advising the human being for the query of any domain from the particular ES.
- **Provide decision-making capabilities:** It provides the capability of decision making in any domain, such as for making any financial decision, decisions in medical science, etc.
- **Demonstrate a device:** It is capable of demonstrating any new products such as its features, specifications, how to use that product, etc.
- **Problem-solving:** It has problem-solving capabilities.
- **Explaining a problem:** It is also capable of providing a detailed description of an input problem.
- **Interpreting the input:** It is capable of interpreting the input given by the user.
- **Predicting results:** It can be used for the prediction of a result.
- **Diagnosis:** An ES designed for the medical field is capable of diagnosing a disease without using multiple components as it already contains various inbuilt medical tools.

Advantages of Expert System

- These systems are highly reproducible.
- They can be used for risky places where the human presence is not safe.
- Error possibilities are less if the KB contains correct knowledge.
- The performance of these systems remains steady as it is not affected by emotions, tension, or fatigue.
- They provide a very high speed to respond to a particular query.

Limitations of Expert System

- The response of the expert system may get wrong if the knowledge base contains the wrong information.
- Like a human being, it cannot produce a creative output for different scenarios.
- Its maintenance and development costs are very high.

- Knowledge acquisition for designing is much difficult.
- For each domain, we require a specific ES, which is one of the big limitations.
- It cannot learn from itself and hence requires manual updates.

Applications of Expert System

- **In designing and manufacturing domain**
  It can be broadly used for designing and manufacturing physical devices such as camera lenses and automobiles.
- **In the knowledge domain**
  These systems are primarily used for publishing the relevant knowledge to the users. The two popular ES used for this domain is an advisor and a tax advisor.
- **In the finance domain**
  In the finance industries, it is used to detect any type of possible fraud, suspicious activity, and advise bankers that if they should provide loans for business or not.
- **In the diagnosis and troubleshooting of devices**
  In medical diagnosis, the ES system is used, and it was the first area where these systems were used.
- **Planning and Scheduling**
  The expert systems can also be used for planning and scheduling some particular tasks for achieving the goal of that task.

S.No.	Human experts	Expert System
1.	Use knowledge in the form of rules of thumbs or heuristics to solve problem in a narrow domain.	It process knowledge expressed in the form of rules and use symbolic reasoning in narrow domain.
2.	In a human expert we deal with human brain in which knowledge exists in a compiled form.	It provide a clear separation of knowledge from its processing.
3.	It is capable of explaining line of reasoning and providing the details.	Expert system helps in tracing the rules that produced during a solving a problem and also explain how a that particular conclusion was reached and why specific data was needed.

S.No.	Human experts	Expert System
4.	It uses inexact reasoning and also able to deal with incomplete, uncertain and fuzzy information.	It permits inexact reasoning and but able to deal with incomplete, uncertain and fuzzy data.
5.	It enhances the quality of problem solving because of years of learning and practical training.	It enhances the quality of problem solving by the addition of new rules or by adjusting the old ones in the knowledge base and when new knowledge is acquired, changes are easy to observe.
6.	Human expert can be available at a specific working day.	Expert system can be available wherever and at any time.
7.	To solve any problem, human expert can take variable time.	To solve any problem, expert system takes a very short interval of time.
8.	It is not replaceable.	It can be replaceable.

Traditional system :

- The traditional system **solves common numerical problems.**
- **It is a sequential program** that combines information and processing.
- A **well-tested program** will never make a mistake.
- There is **no explanation** given for the output.

Expert system :

- It fixes the problem in **a relatively limited area.**
- The **knowledge base may or may not be separated from the processing.**
- It is possible that the **well-tested expert system will make blunders.**
- In most cases, an **explanation is given.**

Applications of Expert Systems

Applications	Role
Design Domain	Camera lens design automobile design
Medical Domain	Diagnosis Systems to deduce the cause of disease from observed dataConduction medical operations on humans.
Monitoring systems	Comparing data continuously with observed systems
Process Control Systems	Controlling physical processes based on the monitoring.
Knowledge Domain	Finding faults in vehicles or computers.
Commerce	Detection of possible fraud Suspicious transactions Stock market trading Airline scheduling, Cargo scheduling.

Conventional System	Expert System
Solves the generic numeric problems.	It solves the problem in very narrow domain.
It is sequential program where information and processing are combined.	The knowledge base is separated from the processing (inference engine). The program may not be sequential.
Tested program never makes mistakes	The well tested expert system may make mistakes and gives wrong answer.
No explanation is provided for output	An explanation is provided in most cases.
When incorrect information is provided, the system may not function.	The system can arrive at a conclusion, even when some information is missing or incomplete.

Issues	Human Expert	Expert System
Availability	Limited	Always
Geographic location	Locally available	Anywhere
Safety considerations	Irreplaceable	Can be replaced
Durability	Depends on individual	Non-perishable
Performance	Variable	High
Speed	Variable	High
Cost	High	Low
Learning Ability	Variable/High	Low
Explanation	Variable	Exact

Aspect	Expert System in AI	Conventional Computer Program
Knowledge Representation	Incorporates domain-specific knowledge and expertise.	Operates based on predefined algorithms and instructions.
Problem-Solving Approach	Utilizes reasoning and inference based on accumulated knowledge.	Follows a fixed set of instructions and algorithms for tasks.
Flexibility and Adaptability	Can adapt and improve with new data or experiences.	Lacks inherent adaptability without manual intervention.
Decision-Making and Reasoning	Provides explanations based on stored rules and knowledge.	Executes tasks without providing reasoning or explanations.
Purpose	Mimics human expertise in specific domains.	Performs diverse tasks based on predefined instructions.
Learning Capability	Can learn and improve performance within its domain.	Typically lacks the ability to learn or improve without coding.
Domain Specificity	Tailored for specific domains where expertise is critical.	Generally applicable across a wide range of tasks and domains.
Examples	Medical diagnosis systems, financial advising systems.	Text editors, games, web browsers, spreadsheet software, etc.

Introduction:
Artificial Intelligence and its applications, Artificial Intelligence Techniques, criteria of success.

(4 hours)

1. **E-Commerce Personalization:** AI tailors product recommendations, chatbot support, and dynamic pricing based on user preferences, enhancing engagement.
2. **Educational Enhancement:** AI facilitates personalized learning, gamification for engaging lessons, and content creation for effective educational experiences.
3. **Robotic Efficiency:** AI enables robots with language understanding, object recognition, and human-robot interaction, improving their decision-making and productivity.
4. **GPS & Navigation:** AI improves navigation systems with voice assistance, intelligent routing, traffic prediction, and precise positioning for efficient travel.
5. **Healthcare Advancements:** AI aids in data analysis for insights, telehealth, patient monitoring, surgical assistance, and disease prediction, revolutionizing healthcare delivery.
6. **Automobile Innovation:** AI powers driving assistance, traffic management, emission reduction, and autonomous driving for safer and efficient transportation.
7. **Agricultural Optimization:** AI monitors crops, manages supply chains, aids in pest control, and predicts weather patterns, enhancing agricultural productivity.
8. **Human Resource Streamlining:** AI automates candidate screening, aids in onboarding, assesses performance, and assists in workforce planning for HR efficiency.
9. **Lifestyle Improvements:** AI drives personalized recommendations, shopping experiences, virtual assistance, and language translation, enhancing everyday life.

10. **Social Media Intelligence:** AI detects fraud, provides insights, conducts sentiment analysis, and moderates content, ensuring a safe and engaging platform.
11. **Gaming Realism:** AI enhances game testing, offers in-game assistance, creates realistic animations, and predicts player behavior for immersive experiences.
12. **Astronomical Analysis:** AI aids in analyzing cosmic data, detecting and classifying astronomical events, and conducting predictive surveys in astronomy.
13. **Chatbot Interaction:** AI-driven chatbots utilize natural language processing (NLP) for interactive and responsive communication with users.
14. **Surveillance & Security:** AI enables object detection, predictive analysis, behavior analysis, and threat detection for enhanced security measures.
15. **Financial Insight:** AI supports fraud detection, risk assessment, and forecasting, aiding in financial planning and secure transactions.
16. **Data Security:** AI detects threats, manages vulnerabilities, and identifies malware, ensuring robust cybersecurity measures.
17. **Travel & Transport Optimization:** AI offers route planning, personalized travel plans, and real-time traffic analysis, improving travel experiences.
18. **Marketing Strategies:** AI segments audiences, creates content, optimizes advertising, and provides insights for effective marketing campaigns.
19. **Entertainment Enhancement:** AI suggests personalized content, analyzes audience behavior, and engages users in real-time, enriching entertainment experiences.
20. **Military & Defense:** AI assists in decision-making, cybersecurity, training simulations, and surveillance for military operational support.

Artificial Intelligence (AI) presents various risks and challenges, including:

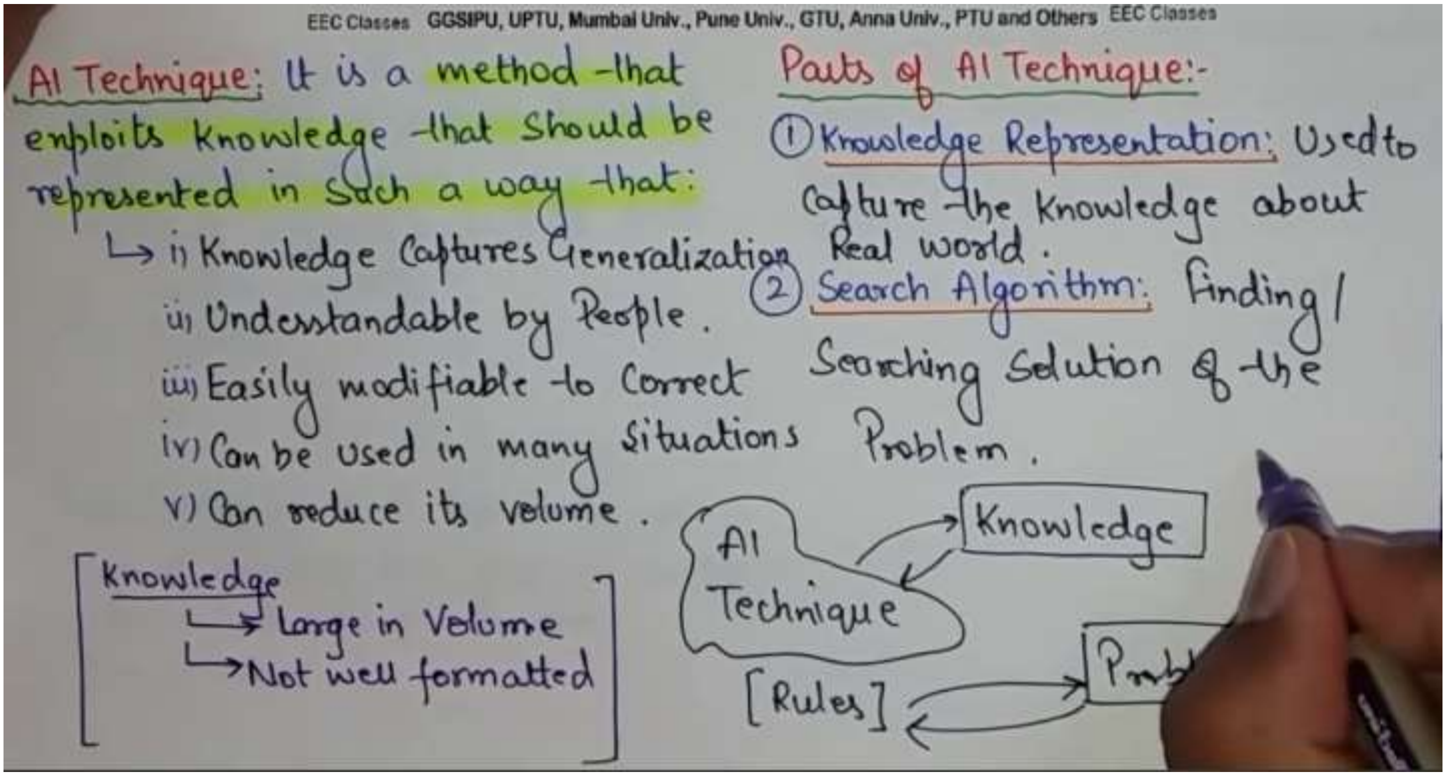
1. **Bias and Fairness:** AI systems can inherit biases present in the data they're trained on, leading to biased outcomes in decision-making, hiring processes, and other critical areas.
2. **Privacy Concerns:** AI often relies on vast amounts of personal data. Improper handling or misuse of this data could lead to privacy breaches and infringements on individual rights.
3. **Job Displacement:** Automation driven by AI could result in the displacement of certain jobs, impacting employment sectors and requiring re-skilling for the workforce.
4. **Security Threats:** AI systems can be vulnerable to attacks and manipulation. If not adequately secured, they may become entry points for cyber threats and breaches.
5. **Lack of Transparency:** Some AI models are complex and lack transparency in their decision-making processes, making it challenging to understand or explain their reasoning.
6. **Ethical Concerns:** As AI becomes more advanced, it raises ethical dilemmas such as the use of AI in weaponry, autonomous vehicles' decision-making in critical situations, etc.
7. **Dependency on AI:** Over-reliance on AI systems without human oversight or intervention might lead to errors or failures that humans are ill-prepared to handle.
8. **Regulatory Challenges:** There's a challenge in creating comprehensive regulations and standards for AI, which can lead to inconsistent or inadequate oversight.

9. **AI Singularity:** Speculation around the potential for AI to surpass human intelligence and control, leading to unpredictable outcomes.
10. **Manipulation and Misuse:** AI can be exploited for malicious purposes, including deepfakes, misinformation, and social engineering attacks.

Applications of AI:-

1) AI in Gaming: Chess, Poker, tic-toe.
↳ machine can think large no. of moves.
2) AI in NLP: Natural lang. Processing
↳ Machine can understand human lang.
3) AI in HealthCare: Fast diagonsis
↳ Robotic Surgery.
4) AI in Finance: Adaptive Intelligence.
↳ automatic chatbots, algorithm trading.
5) AI in Data Security: Helps in making data/applⁿ more Secure.
↳ AEG bot, AI2
6) Expert System: Integration of s/w, machine and Special infoⁿ to provide reasoning & advise.
7) Computer Vision: Understand the visual automatically by machine.
8) Speech Recognition: Extract the meaning of sentence by human talk. [Slang removal, noise rem.-]
9) Robotics:
↳ Erica and sophia.
Talk and behave like humans.
10) AI in e-Commerce: Automatic

Advantages of AI	Disadvantages of AI
Accuracy ↑ & Error ↓	COST ↑
Fast Decision Making.	Can't thing beyond th limits.
Reliability is more	No feeling & emotions
Usefulness in Risky Area.	more dependency on machines ↑.
Digital Assistant	No original thinking



Artificial Intelligence (AI) encompasses a range of techniques and methodologies used to enable machines to simulate human intelligence. Here are some fundamental AI techniques:

1. **Machine Learning (ML):** ML allows machines to learn from data and experiences without being explicitly programmed. It includes:
   - **Supervised Learning:** Training models with labeled data to make predictions or classifications.
   - **Unsupervised Learning:** Extracting patterns or structures from unlabeled data.
   - **Reinforcement Learning:** Training models to make sequences of decisions through trial and error, based on rewards or penalties.
2. **Deep Learning:** A subset of ML that uses artificial neural networks with multiple layers to learn complex patterns and representations. It powers advancements in image and speech recognition, natural language processing, and more.
3. **Natural Language Processing (NLP):** NLP enables machines to understand, interpret, and generate human language. Techniques include sentiment analysis, language translation, named entity recognition, and text generation.
4. **Computer Vision:** This field focuses on enabling machines to interpret visual information from images or videos. Techniques involve object detection, image classification, segmentation, and facial recognition.
5. **Expert Systems:** These are AI systems that leverage knowledge representations, inference engines, and rule-based reasoning to emulate human expertise in specific domains. They're used for decision-making and problem-solving.
6. **Fuzzy Logic:** A technique that handles reasoning and decision-making in an uncertain or imprecise environment by considering degrees of truth rather than binary values (true or false).
7. **Genetic Algorithms:** Inspired by the process of natural selection, these algorithms mimic biological evolution to solve optimization and search problems by generating high-quality solutions.

8. **Probabilistic Reasoning:** Utilizing probability theory to model uncertainty and make decisions in situations where outcomes are uncertain.
9. **Swarm Intelligence:** Modeling the collective behavior of decentralized, self-organized systems (inspired by the behavior of social insects like ants or bees) to solve complex problems.
10. **Robotics:** Combining AI with robotics to create intelligent systems that can sense, reason, and act in the physical world. Techniques involve sensor data processing, motion planning, and control systems.

Success in artificial intelligence (AI) can be measured by several criteria:

Turing test

1. **Accuracy and Performance:** AI systems should demonstrate high accuracy in their predictions, classifications, or tasks. They should perform efficiently, delivering results within acceptable time frames.
2. **Robustness and Reliability:** Successful AI models should perform well in various conditions and scenarios, exhibiting robustness against noise, outliers, or unexpected inputs. They should also be reliable, consistently providing accurate results.
3. **Scalability:** The ability of AI systems to handle increasing amounts of data, users, or complexity without a significant decrease in performance is crucial. Scalability ensures that the AI remains effective as demands grow.
4. **Interpretability and Transparency:** Being able to understand and interpret how an AI system arrives at its decisions or recommendations is essential for trust and accountability. Transparent AI models are easier to explain and troubleshoot.
5. **Ethical and Fair Use:** Success in AI involves ensuring that AI applications are used ethically and without bias. Fairness in decision-making and mitigation of biases are critical factors for success.
6. **Adaptability and Continual Learning:** AI systems that can adapt to new data, learn from experiences, and improve over time without human intervention demonstrate success. Continual learning ensures that the AI remains relevant and up-to-date.
7. **User Acceptance and Usability:** Successful AI solutions should be user-friendly, providing a seamless experience for users and meeting their needs effectively.
8. **Cost-effectiveness:** AI systems that provide significant benefits while managing costs efficiently are considered successful. They should offer value relative to the resources invested.
9. **Regulatory and Legal Compliance:** Compliance with regulations, ethical standards, and legal frameworks is crucial for the successful deployment and use of AI systems.
10. **Impact and Value Creation:** Ultimately, successful AI should create tangible value, whether in terms of improved efficiency, better decision-making, cost savings, or societal benefits.

S. No.	Feature	Artificial Intelligence	Human Intelligence
1.	Emergence	AI is an advancement made by human insights; its early improvement is credited to Norbert Weiner who theorized on criticism mechanisms.	On the other hand, human creatures are made with the intrinsic capacity to think, reason, review, etc.
2.	Nature	Artificial intelligence (AI) strives to build machines that can mimic human behavior and carry out human-like tasks.	Human intelligence seeks to adapt to new situations by combining a variety of cognitive processes.
3.	State	Machines are digital.	The human brain is analogous.
4.	Function	AI-powered machines rely on input of data and instructions.	Humans use their brains' memory, processing power, and cognitive abilities.
5.	Pace/Rate of AI and human	As compared to people, computers can handle more data at a speedier rate. For occurrence, in the event that the human intellect can solve a math problem in 5 minutes, AI can solve 10 problems in a minute.	In terms of speed, humans cannot beat the speed of AI or machines.
6.	Learning ability	As machines are unable to reason abstractly or draw conclusions from the past. They can only acquire knowledge through information and frequent training, but they will never develop a human-specific thinking process.	Learning from various events and prior experiences is the foundation of human intelligence.
7.	Decision Making	AI is profoundly objective in choice making because it analyzes based on absolutely accumulated data.	Human choices may be affected by subjective components which are not based on figures alone.
8.	Perfection	AI frequently produces precise comes about because its capacities are based on a set of modified rules.	For human insights, there's more often than not room for "human error" as certain subtle elements may be missed at one point or the other.

S. No.	Feature	Artificial Intelligence	Human Intelligence
9.	Energy Consumption	The modern computer generally uses 2 watts of energy.	On the other hand, human brains uses about 25 watts
10.	Modification of AI and Human	AI takes much more time to adjust to unused changes.	Human insights can be adaptable in reaction to the changes in their environment. This makes individuals able to memorize and ace different skills.
11.	Versatility	AI can as it were perform fewer assignments at the same time as a framework can as it were learn duties one at a time.	The human judgment skills underpin multitasking as proven by differing and concurrent roles.
12.	Social Networking	AI has not aced the capacity to choose up on related social and enthusiastic cues.	On the other hand, as social creatures, people are much way better at social interaction since they can prepare theoretical data, have self-awareness, and are delicate to others' feelings.
13.	Task	It does optimization of the system. It cannot be creative or innovative as humans can only think and machines cannot.	It is innovative or creative.

	Artificial Intelligence	Human Intelligence
Processing	Based on algorithms and mathematical models	Based on cognitive processes and biological structures
Learning	Based on data and feedback loops	Based on experience, intuition, and creativity
Speed	Can process data and perform tasks much faster than humans	Slower than AI in processing large amounts of data, but can make complex decisions quickly

Adaptability	Can quickly adapt to new data and situations	Can adapt to new situations, learn from experience, and make decisions based on context
Emotions	Lacks emotions and empathy	Capable of feeling emotions and empathy
Creativity	Limited ability to be creative or think outside of the box	Capable of creativity, imagination, and innovation
Ethics	Does not have a moral code or conscience	Has a moral code and conscience that guides decision-making
Physical Limitations	Does not have physical limitations, can operate 24/7	Limited by physical capabilities and requires rest and maintenance

Artificial Intelligence (AI)	Human Intelligence
Makes Objective Decisions	Makes Efficient Decisions for Ambiguous and Uncertain Cases
Fast Calculation Abilities	Works Better in Novel Situations
Able to Mimic Human Conversations	Emotional and Social Intelligence
Low Chances of Errors in Familiar Problems	Abstract Thinking Ability
Able to Conduct Research and Hunt Facts Within Few Seconds	Ability to Look Ahead to Future

a) Define artificial intelligence. Explain the necessary components to define an AI problem with example.

Defining an artificial intelligence (AI) problem involves several essential components to ensure clarity and effectiveness in solving it. Here are the necessary components:

1. **Problem Statement:** Clearly articulate the problem to be solved. It should be specific, well-defined, and focused. For example, "Develop an AI system to predict customer churn in a telecommunications company."
2. **Objective:** Define the desired outcome or goal of solving the problem. It could be maximizing accuracy, minimizing error, optimizing a process, or achieving a specific performance metric. For instance, "Achieve at least 85% accuracy in predicting customer churn."

3. **Data Requirements:** Specify the data needed to address the problem. This includes the type of data (structured, unstructured), sources, size, format, and whether labeled or unlabeled data is necessary. For instance, "Access customer demographics, usage history, and churn status."
4. **Feature Selection/Engineering:** Identify the relevant features or variables within the data that are essential for solving the problem. Feature engineering may involve transforming or selecting features that have the most predictive power.
5. **Evaluation Metric:** Determine how the performance of the AI system will be measured. Common metrics include accuracy, precision, recall, F1-score, or area under the curve (AUC), depending on the nature of the problem.
6. **Model Selection:** Choose the appropriate AI technique or model(s) to solve the problem. This could involve selecting from various machine learning algorithms, deep learning architectures, or other AI methodologies.
7. **Training and Testing Strategy:** Define how the model will be trained on the data (e.g., cross-validation, train-test split) and how its performance will be validated or tested.
8. **Constraints and Assumptions:** Identify any constraints, limitations, or assumptions related to the problem. For instance, computational resources, time constraints, data privacy concerns, or assumptions about the data distribution.
9. **Ethical and Societal Considerations:** Consider ethical implications or societal impacts of solving the problem, such as fairness, bias, privacy, or potential consequences of the AI system's decisions.
10. **Iterative Process:** Recognize that AI problem-solving is often iterative. It may involve multiple rounds of model refinement, feature engineering, and evaluation before achieving the desired outcome.

By outlining these components clearly and comprehensively, you establish a robust framework to tackle AI problems systematically, ensuring a more effective and structured approach to finding solutions.

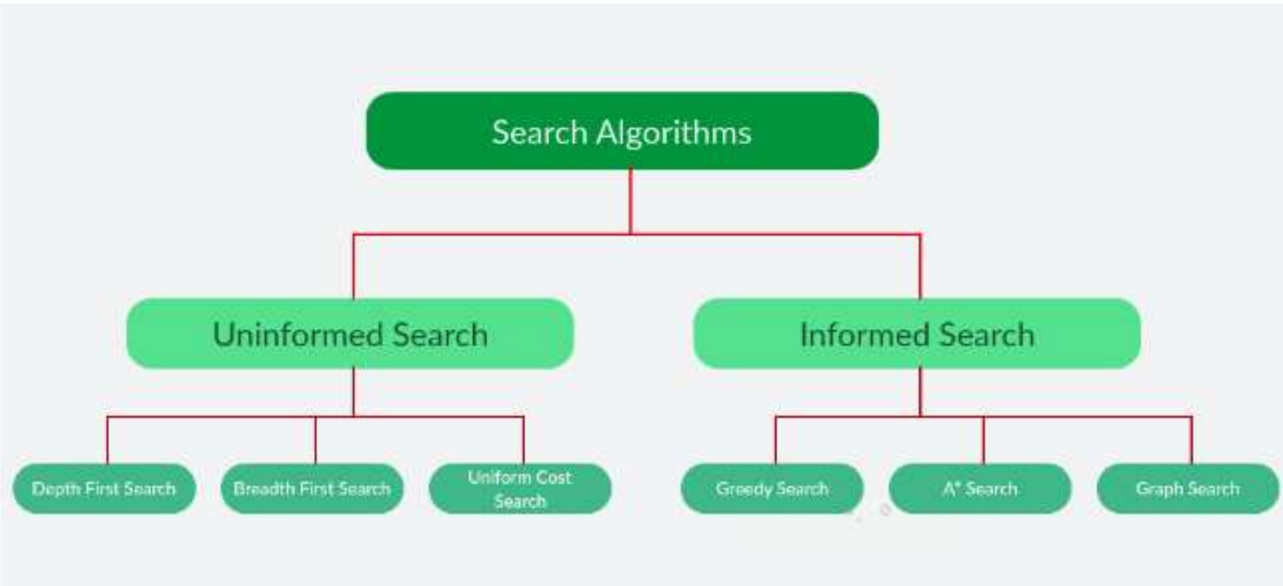
Let's define an artificial intelligence problem using an example in the context of customer churn prediction for a telecommunications company:

1. **Problem Statement:** Develop an AI system to predict customer churn in a telecommunications company.
2. **Objective:** Achieve at least 85% accuracy in predicting customer churn to proactively identify and retain potentially leaving customers.
3. **Data Requirements:** Access customer demographics, usage history, and churn status. Gather data on customer age, gender, location, tenure, service usage, call logs, contract details, and whether the customer churned or not.
4. **Feature Selection/Engineering:** Identify relevant features such as customer tenure, call duration, service plan details, customer complaints, and frequency of usage. Transform categorical variables into numerical representations (e.g., one-hot encoding).
5. **Evaluation Metric:** Use accuracy, precision, recall, and F1-score to evaluate the model's performance in predicting churn. Prioritize high recall to capture as many actual churn cases as possible.
6. **Model Selection:** Choose appropriate machine learning models like logistic regression, decision trees, random forests, or a neural network for prediction based on the dataset's characteristics.
7. **Training and Testing Strategy:** Split the dataset into training and testing sets (e.g., 70/30 or using cross-validation). Train the chosen models on the training data and evaluate their performance on the test set.
8. **Constraints and Assumptions:** Consider computational resources, time constraints for model training and testing, and the assumption that historical data patterns will continue in the future.
9. **Ethical and Societal Considerations:** Ensure fairness in predictions, mitigate biases, and maintain customer data privacy and confidentiality throughout the process.
10. **Iterative Process:** Refine the models by adjusting hyperparameters, exploring additional features, or addressing issues like class imbalance. Iterate through model training, evaluation, and refinement until achieving the desired performance.

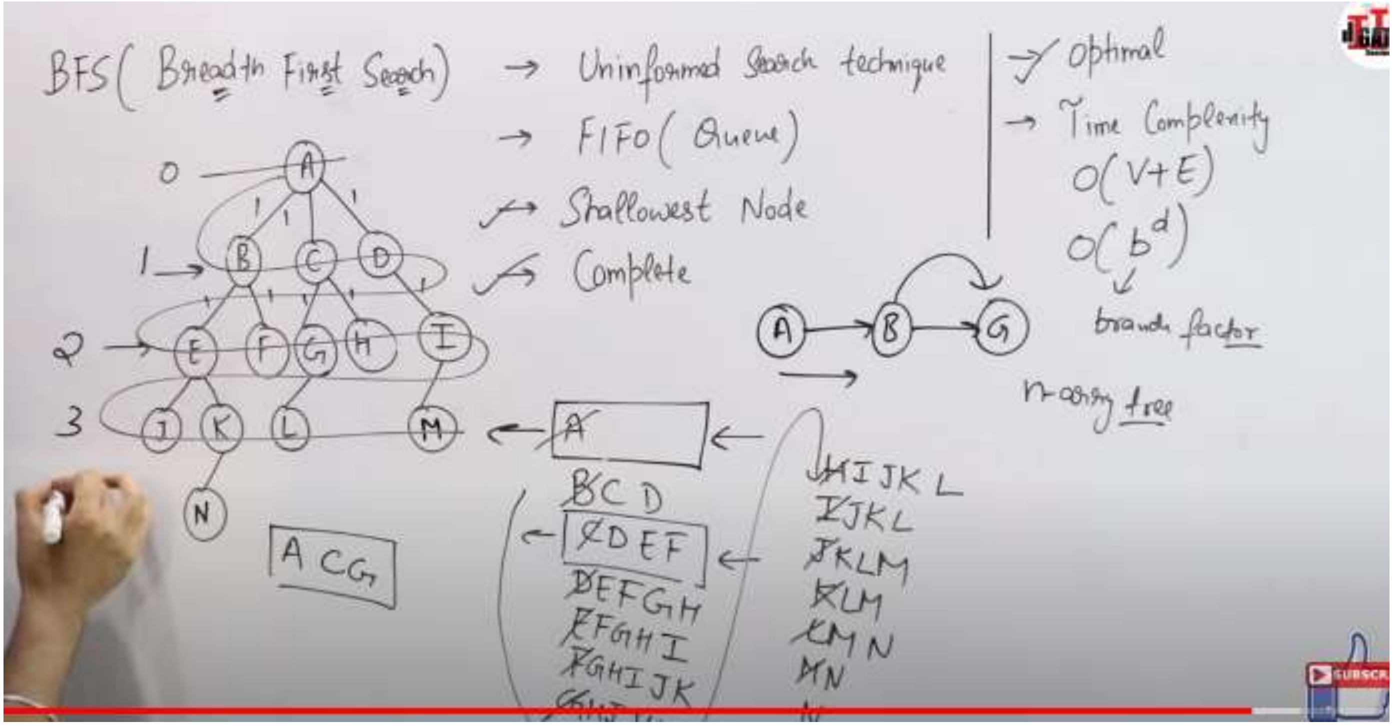
By defining these components clearly, the telecommunications company can systematically approach the customer churn prediction problem, select appropriate AI techniques, and evaluate the model's effectiveness in reducing churn rates.

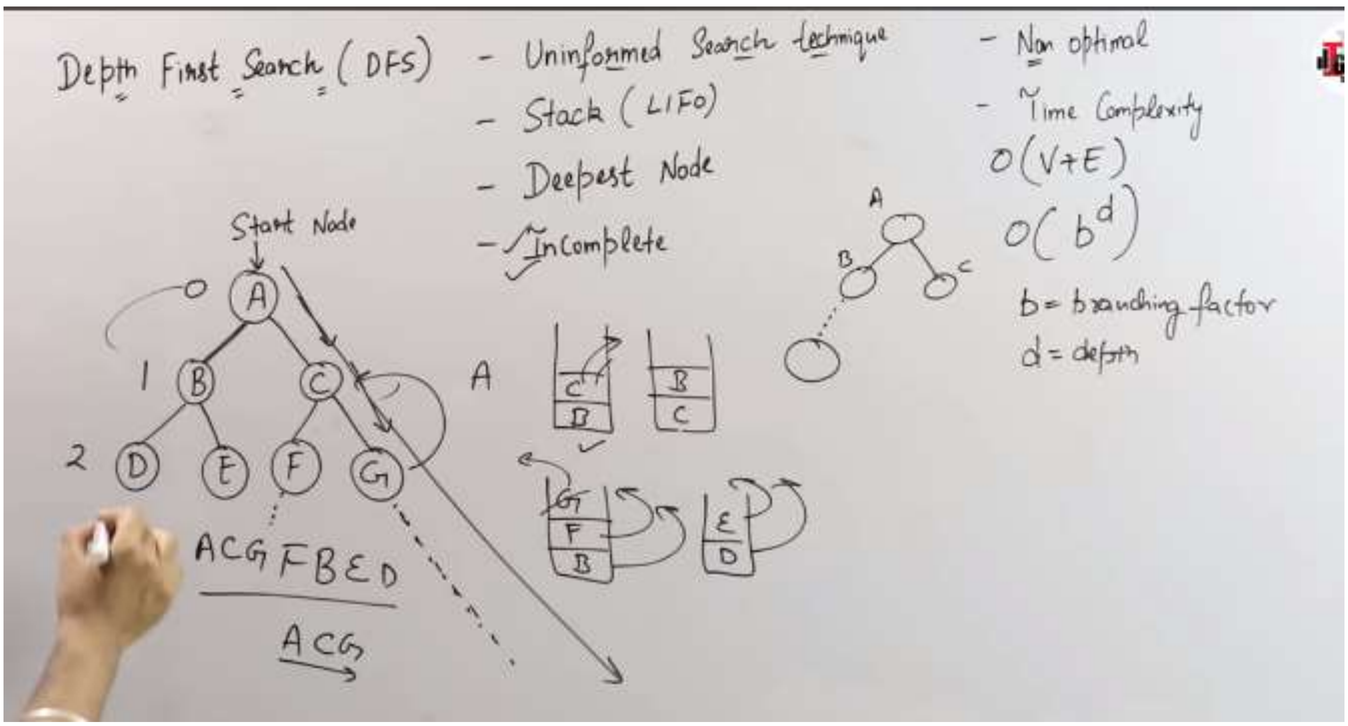
Problem solving techniques:
State space search, control strategies, heuristic search, problem characteristics, production system characteristics., Generate and test, Hill climbing, best first search, A* search, AO* search, Constraint satisfaction problem, Agenda Driven Search, Mean-end analysis, Min-Max Search, Alpha-Beta Pruning, Iterative Deepening.

(9 hours)



Uninformed Searching	Informed Searching
) Search without Information	1) Search with information
) No knowledge	2) Use knowledge to find steps to solution
) Time Consuming	3) Quick solution
) More Complexity (Time, Space)	4) Less Complexity (Time, Space)
) DFS, BFS etc.	5) A*, Heuristic DFS, Best first Search


BFS (Breadth First Search)
→ Uninformed Search technique
→ FIFO (Queue)
→ Shallowest Node
→ Complete
→ Optimal
→ Time Complexity
O(V+E)
O(b^d)
branch factor
n-ary tree
A CG
BCD
CDEF
DEFGH
EFGHI
FGHIJK
HIJKL
IJKL
JKLM
KLM
LMN
MN


Depth First Search (DFS)
- Uninformed Search technique
- Stack (LIFO)
- Deepest Node
- Incomplete
- Non optimal
- Time Complexity
O(V+E)
O(b^d)
b = branching factor
d = depth
Start Node
ACGFBED
ACG

Best First Search (Informed, Heuristic)

Algorithm:

→ Let OPEN be a priority queue containing initial state.

Loop

if OPEN is empty return failure

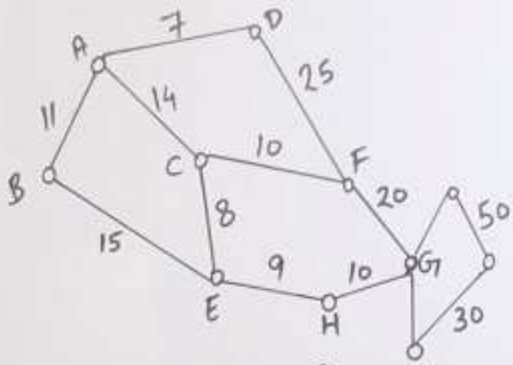
Node ← Remove-First (OPEN)

if Node is a Goal

then return the Path from initial to Node

else generate all successors of Node and Put the newly generated Node into OPEN according to their f values

END LOOP

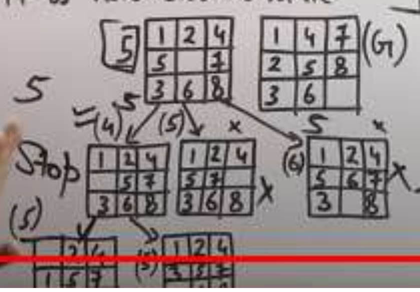


Straight line dist.

A → G = 40
B → G = 32
C → G = 25
D → G = 35
E → G = 19
F → G = 17
H → G = 10
G → G = 0

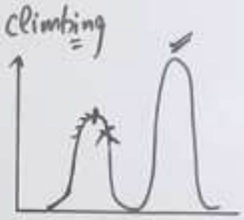
"Simple Hill Climbing Algorithm" (Local search Algo, greedy approach, No backtrack)

1. Evaluate the initial state
2. Loop Until a solution is found or there are no operators left
   - Select and Apply a new operator
   - Evaluate the new state:
     if goal then quit
     if better than current state then it is new current state.



Problems in Hill climbing

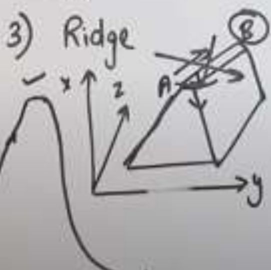
1) Local Maximum

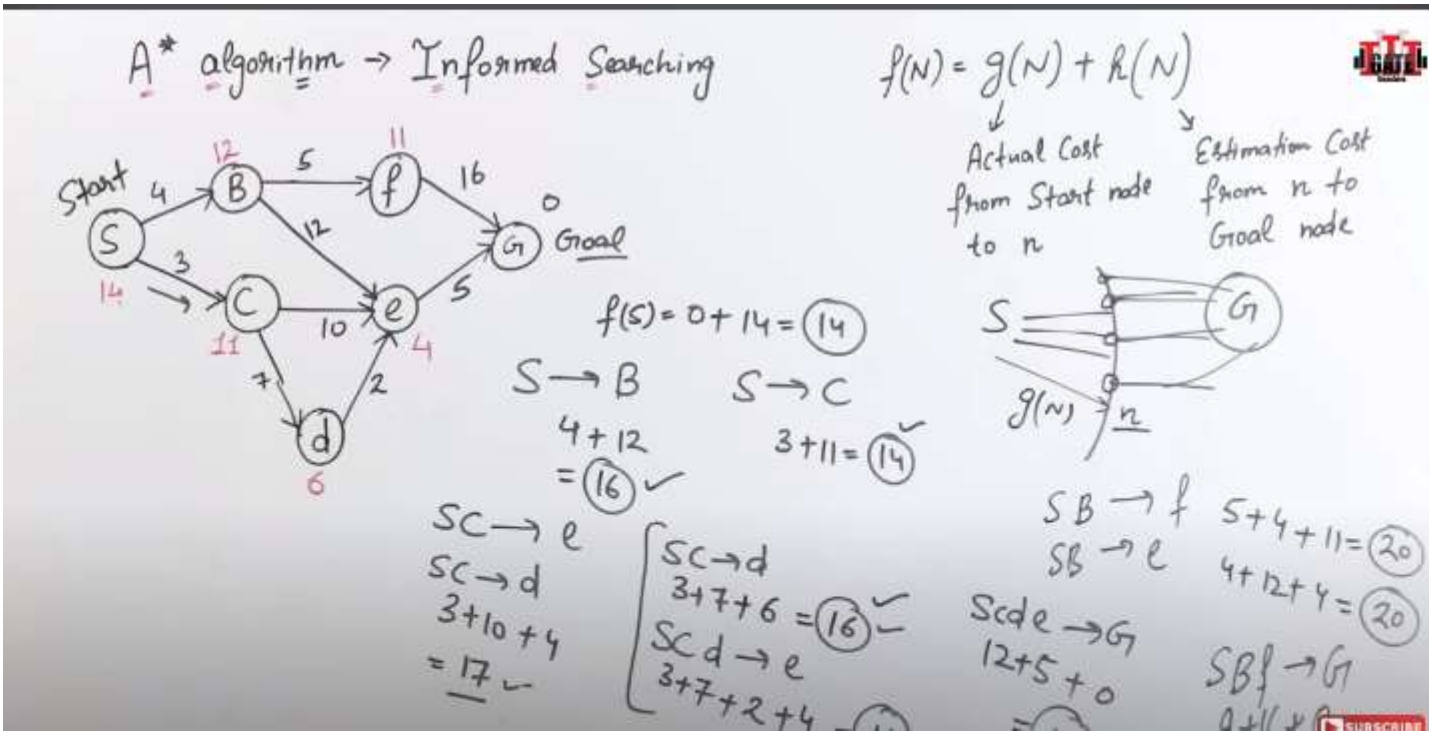


2) Plateau/Flat Maximum

Stop

3) Ridge




A* algorithm → Informed Searching
f(N) = g(N) + h(N)
Actual Cost from Start node to n
Estimation Cost from n to Goal node
Start
Goal
f(S) = 0 + 14 = 14
S → B
4 + 12 = 16
S → C
3 + 11 = 14
SC → e
SC → d
3 + 10 + 4 = 17
SC → d
3 + 7 + 6 = 16
SCd → e
3 + 7 + 2 + 4
SB → f
SB → e
5 + 4 + 11 = 20
4 + 12 + 4 = 20
Scde → G
12 + 5 + 0
SBf → G
g(N)
n

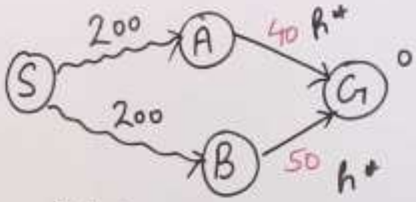
How to make A* admissible:

$$\boxed{\underset{\text{estimated}}{h(n)} \le \underset{\text{Actual}}{h^*(n)} \text{ - underestimation}}$$

$h(n) \ge h^*(n)$ - Overestimation (optimal)

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

$= 240 + 0$



$g(G)$ =

$S \xrightarrow[g(n)]{} n \xrightarrow{h} G$ $\quad f(n) = g(n) + h(n) = 240$

$g(A) = 200$

$g(B) = 200$

Case I overestimation

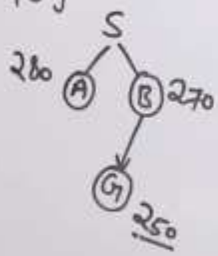
$\left.\begin{array}{l} h(A) = 80 \\ h(B) = 70 \end{array}\right] > h^*$

$f(A) = 200 + 80 = 280$

$f(B) = 200 + 70 = 270$

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

$250 + 0 = \underline{250}$



Case II Underestimation

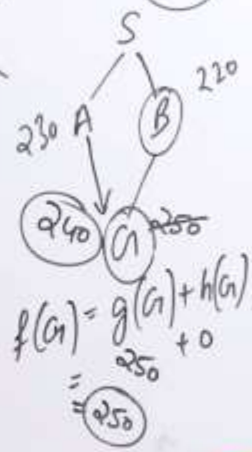
$h(A) = 30$

$h(B) = 20$

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

$= 200 + 30 = 230$

$f(B) = 200 + 20 = 220$



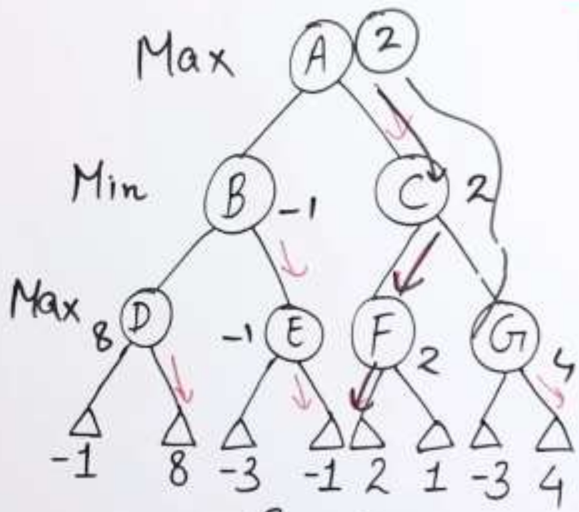
$f(G) = g(G) + h(G)$

$= 250 + 0$

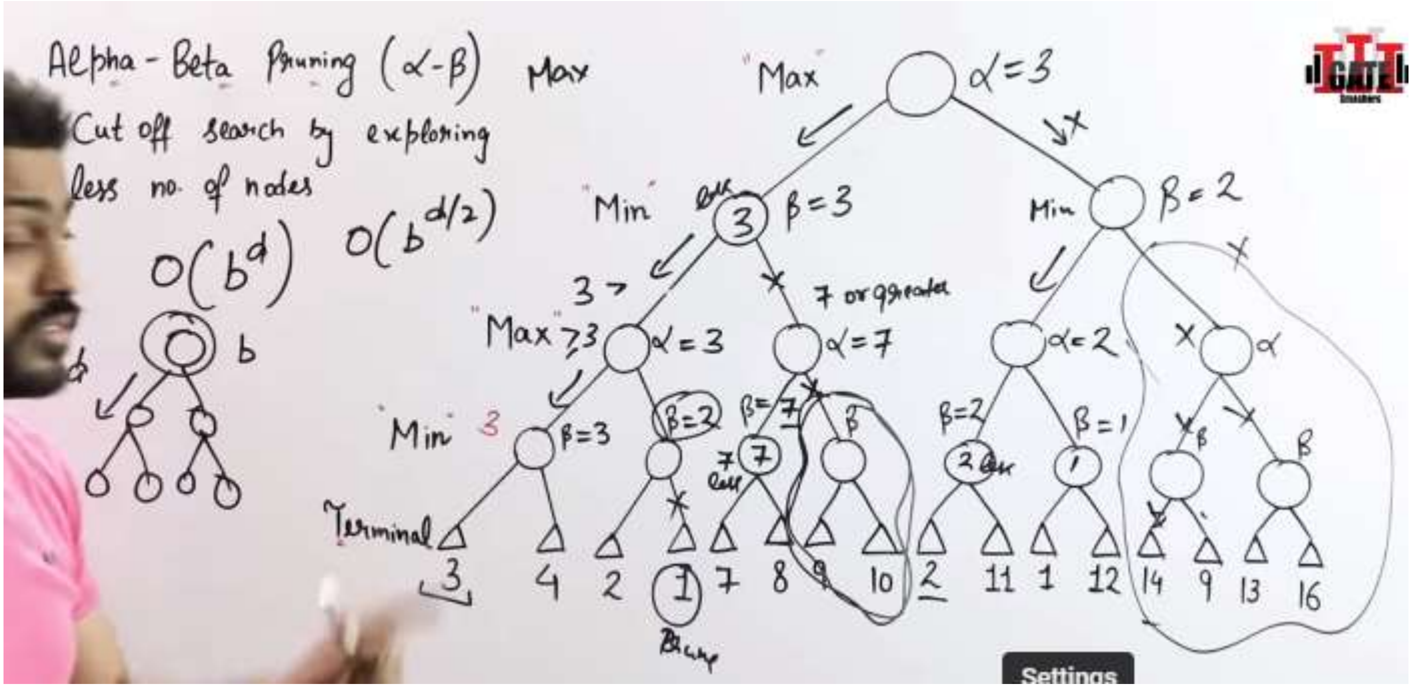
$= 250$

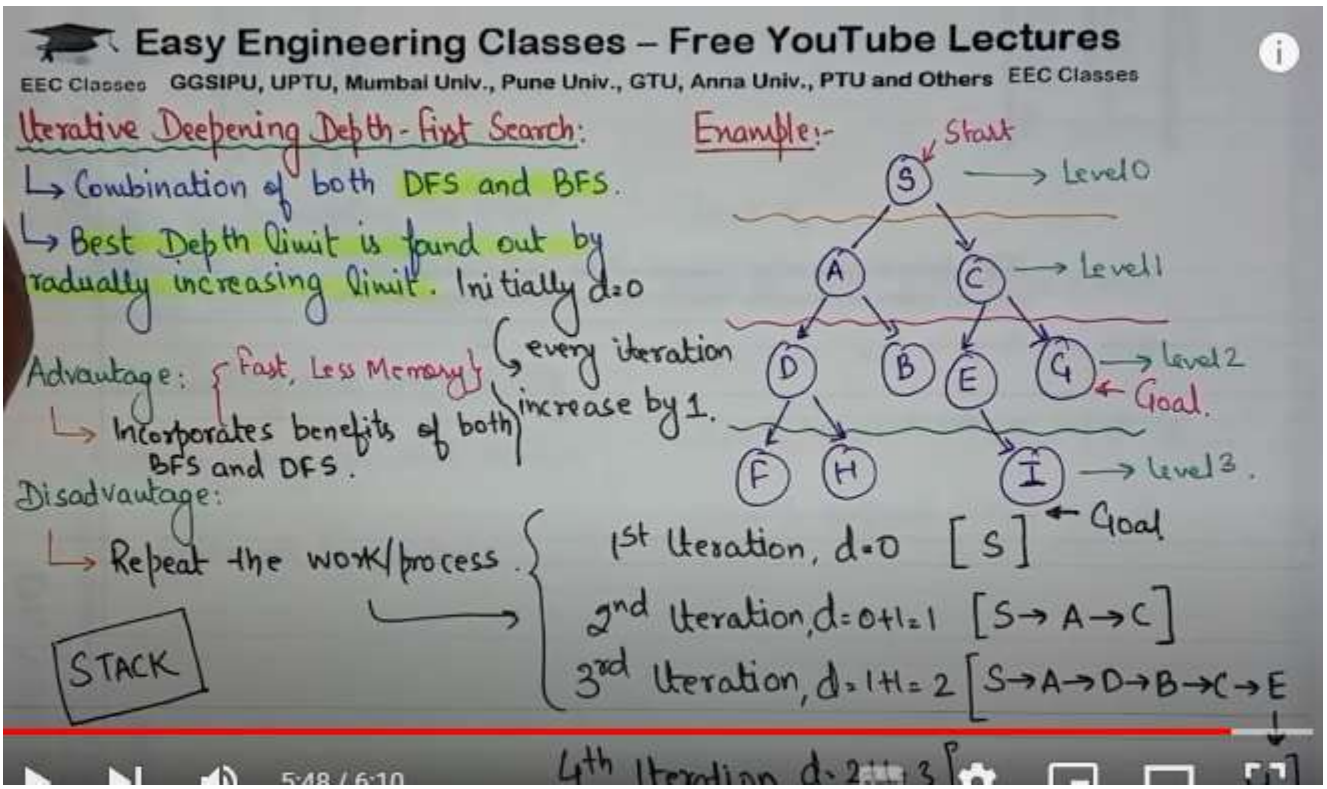
Minimax Algorithm

- ⇒ * Backtracking algorithm
- ⇒ Best move strategy used
- ⇒ Max will try to maximize its utility (Best Move)
- → Min will try to minimize utility (Worst move)



Game Tree


Alpha - Beta Pruning (α-β)
Max
Cut off search by exploring
less no. of nodes
O(b^d)
O(b^{d/2})
Max
α=3
Min
β=3
Min
β=2
7 or greater
Max ≥3
α=3
α=7
α=2
α
Min
β=3
β=2
β=7
β
β=2
β=1
β
β
Terminal
3 4 2 1 7 8 9 10 2 11 1 12 14 9 13 16
Settings


Easy Engineering Classes – Free YouTube Lectures
EEC Classes GGSIPU, UPTU, Mumbai Univ., Pune Univ., GTU, Anna Univ., PTU and Others EEC Classes
Iterative Deepening Depth-First Search:
↳ Combination of both DFS and BFS.
↳ Best Depth limit is found out by gradually increasing limit. Initially d=0
every iteration increase by 1.
Advantage: {Fast, Less Memory}
↳ Incorporates benefits of both BFS and DFS.
Disadvantage:
↳ Repeat the work/process
STACK
Example:-
Start
S → Level 0
A C → Level 1
D B E G → Level 2
Goal.
F H I → Level 3.
Goal
1st Iteration, d=0 [S]
2nd Iteration, d=0+1=1 [S→A→C]
3rd Iteration, d=1+1=2 [S→A→D→B→C→E
4th Iteration
5:48 / 6:10

Ques:- Differentiate b/w blind and heuristic Search.

Blind Search:- It is also Known as unknown/uninformed Search.
- ↳ There is no info^n about the Searching.
- ↳ No Knowledge of where the GOAL.
- ↳ Eg:- Depth first, Breadth first Search
- ↳ Efficiency is low
- ↳ Slower than Heuristic
- ↳ Large memory is used
- ↳ No func^n (Special) is used.

Heuristic Search:- It is a method of Solving problems more easily and fast. They have Knowledge of where goal or finish of the graph. (Informed Search)

Eg:- Hill climbing, A*, AO*
- ↳ Highly efficient < less time, Less cost
- ↳ finds sol^n Quickly.
- ↳ no large memory is required.
- ↳ Heuristic func^n is used.

State Space Search: Used in Problem Solving.

↳ It is a process used in A.I in which successive Configurations or States of an instance are considered with intention of finding a GOAL State with desired property.

↳ Problems are modelled as State Space

↳ Representation:

S: (S, A, Action(S), Result (S,a), Cost(s,a)) } Set of states in which a problem Can be.

- S → set of all possible States.
- A → Set of all possible actions
- Action(S) → Func^n [which action is possible for Current state]
- Result (S,a) → Func^n [State reached by performing action 'a' on state 's']
- Cost(s,a) → Costing.

EIGHT TILE PUZZLE

Start (S)

1	4	3
2		5
8	6	7

GOAL (G)

1	2	3
8		4
7	6	5

Actions Possible:- → up, → down, → Left, → Right } Legal moves for a State

'Right'

1	4	3
2	5	
8	6	7

} right is not possible now

→ up
→ down
→ left

(AI 40) Easy Engineering Classes – Free YouTube Lectures
EEC Classes GGSIPU, UPTU, Mumbai Univ., Pune Univ., GTU, Anna Univ., PTU and Others EEC Classes

Ques:- Describe a Production System in AI.

↳ Helps in Structuring AI Programs in a way that facilitates describing and performing the Search Process.

Production System Consists of:-
- (i) Set of Rules
- (ii) Knowledge Base
- (iii) Control Strategy
- (iv) Rule Applier.

Steps to Solve the Problem:-

↳ i) First reduce Problem so that it can be shown in a precise statement.
ii) Problem can be solved by Searching a path through Space. [Start → Goal]
iii) Solving process can be modelled.

Advantages:-

(i) Structuring AI Problems.] Excellent tool.
(ii) Highly modular.] rules – add, remove, change.
(iii) Rules are expressed in natural form] Easy to understand.

Characteristics:-

↳ i) Monotonic PS: Appl^n of a rule never prevents later appl^n of another rule.] Rules are Independent.
ii) Non-Monotonic PS: which this is not true.
iii) Partially Commutative PS: (r1, r2) $x \to y$ Permutation $(x \to y)$
iv) Commutative PS: Both monotonic and Partially Commutative.

"Knowledge Representation and Reasoning"

- → Logic → Propositional logic, Predicate logic
- → Rules → if then
- → Semantic Net → GoogleGraph
  Meaning graph
- → Frame → Slots and fillers
  Object Attribute
- → Script

Syntax

Semantic

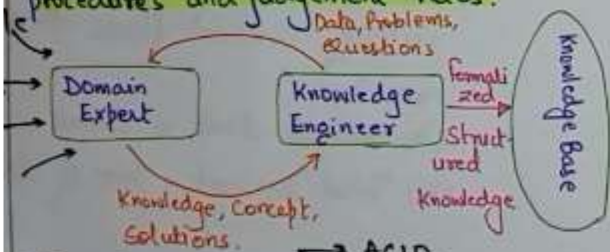
(AI65) KA TECHNIQUES:

Ques:- Define the term Knowledge Acquisition (KA)? Describe the techniques used.

Human expert. Subset = Knowledge elicitation

'KA' is the process of extracting, structuring and organizing knowledge from one or more sources.

→ Knowledge is a collection of specialized facts, procedures and judgement rules.



Tasks in KA: (CIAD) → ACID

↳ i) Collect :- Acquiring knowledge from expert.

ii) Interpret :- Review & identifying key parts.

iii) Analyze :- Forming theories, strategies.

iv) Design :- Forming better understanding of problem

(i) Introspection: Expert acts as both expert and knowledge engineer.

(ii) Observation: expert closely observed work.] video recording, Analysis.

(iii) Induction: Converting set of examples into rules.

(iv) Protocol Analysis: (Psychology)

↳ expert is asked to perform task and verbalized thought process.

(v) Prototyping: extension of interviewing.

expert + K.E ⇒ System] expert $\xrightarrow{\text{test}}$ K.E, modification

(vi) Interviewing:

(vi) Interviewing:

↳ Early stages of acquisition.

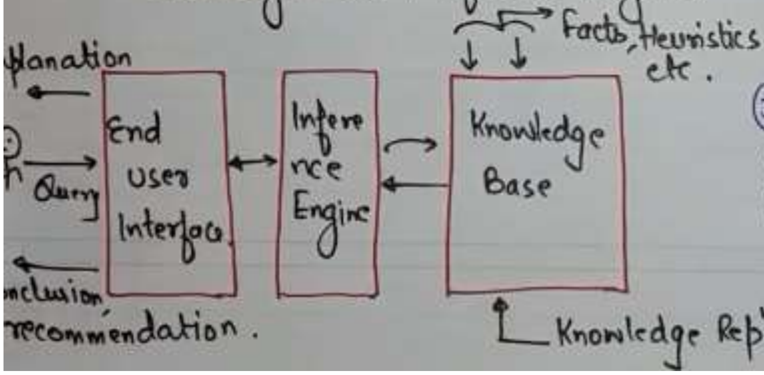
↳ experts verify the knowledge.

Architecture of Knowledge Based System:

'KBS' is a system that draws upon knowledge of Human Experts captured in Knowledge Base to solve problems that normally require human expertise.

↳ KBS is more general than Expert System.

↳ Knowledge Base + Inference Engine.



Components of KBS:

① Knowledge Base: Contains system organized collection of facts about a system domain.

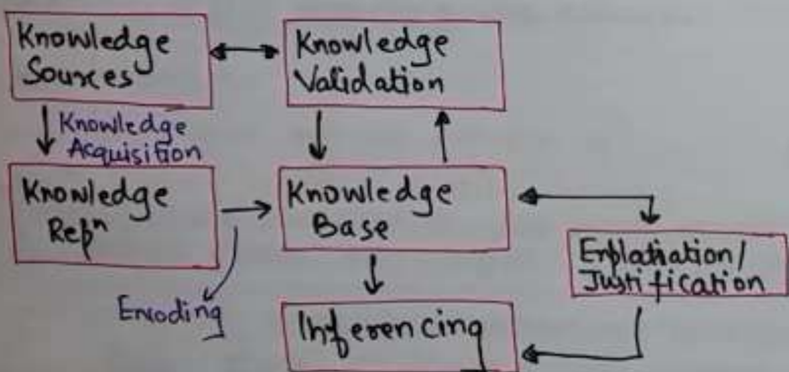
② Inference Engine: Tries to derives answers from Knowledge Base. [BRAIN]

③ User Interface: Enables the user to communicate with KBS.

Ques:- What is/are the role(s) of Knowledge Engineers? Also explain difficulties in Knowledge acquisition.

Knowledge Engineering is the process of digitizing the Knowledge and utilising it to resolve complicated problems. ↳ Knowledge is an essential part of AI.

Process:-



ROLE OF Knowledge Engineer:- They develop these s/w or systems by gathering all relevant infon and putting it into Systematic format.

↳ i) Problem Assessment

ii) Knowledge Structure development.

iii) Incorporation of Infon into Structure.

iv) Testing and Validating Infon.

v) System Maintenance

Easy Engineering Classes – Free YouTube Lectures
EEC Classes GGSIPU, UPTU, Mumbai Univ., Pune Univ., GTU, Anna Univ., PTU and Others EEC Classes

Knowledge Acquisition Difficulties:- → Communication Gap. → Complex activity.

↳ ① People find it difficult to

- ↳ i) Know exactly what person wants.
- ↳ ii) Give right amount of details.
- ↳ iii) Presenting ideas in clear form.
- ↳ iv) Explain all terminology.

② People attempting to acquire Knowledge from someone find it difficult to

- ↳ i) Understand everything a person saying.
- ↳ ii) Note/write down everything.
- ↳ iii) maintaining level of understanding.
- ↳ iv) fully domain specific.

Types of Knowledge:-

① **Procedural Knowledge:-** Describes How to Solve Problem.

- ↳ Also Known as Imperative Knowledge.
- ↳ Provides dirⁿ on how to do Something.

} Procedure to do something.

- ↳ Can be directly applied to a task.
- ↳ Includes → Rules → Strategies → procedures ...

② **Declarative Knowledge:-** Describes what is Known about problem

what

- ↳ Tells us facts - what things are.
- ↳ Includes → Concepts → Facts
- ↳ Also called descriptive knowledge.

③ **Meta-Knowledge:-** Describes knowledge about another knowledge.

- ↳ used to pick other knowledge that is best suited for solving a problem.
- ↳ Knowledge about other types of Knowledge

④ **Heuristic Knowledge:-** It is representing Knowledge of Some experts in a field or Subject.

- ↳ describes rules of thumb that guides reasoning process.
- ↳ Prev. experiences, approaches ...

Easy Engineering Classes – Free YouTube Coaching
For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India

5) Structural Knowledge:- It is the basic knowledge to problem solving.

↳ describes relationships b/w Various Concepts/objects.
- Kind of
- Part of
- group of

↳ describes an expert overall mental model of a problem.

EEC Classes GGSIPU, UPTU, Mumbai Univ., Pune Univ., GTU, Anna Univ., PTU and Others EEC Classes

Ques:- What are Semantic Networks? Explain with the help of an example.

↳ Graphical notation for representing knowledge in interconnected nodes pattern.

↳ Popular in AI and NLP ∵ it represents knowledge or support reasoning.

↳ Alternative of Predicate Logic

↳ Nodes represents objects
- circle (Physical)
- Ellipse (Concept)
- Rectange (Situation)

↳ Arcs represents Relⁿ b/w objects] Links/arrows

↳ Link Labels] Specify relationships

↳ Also known as Associative Nets] nodes are associated.

COMPONENTS:

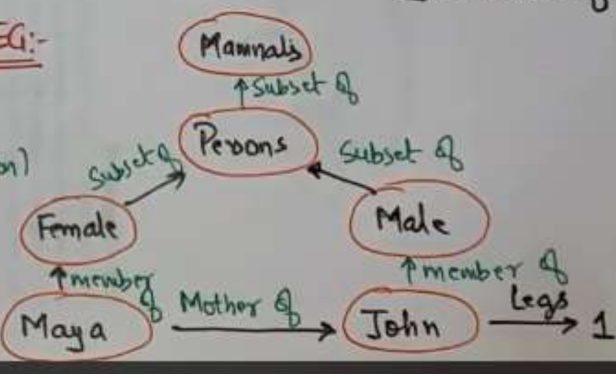
① Lexical Components. Nodes, Links, Labels.

② Structural: Link or Nodes] Directed.

③ Semantic:] Defⁿ related to links/nodes.
↳ Facts.

④ Procedural:
- Constructor] creation of new links
- Destructor] removal of link.

EG:-



Easy Engineering Classes – Free YouTube Coaching

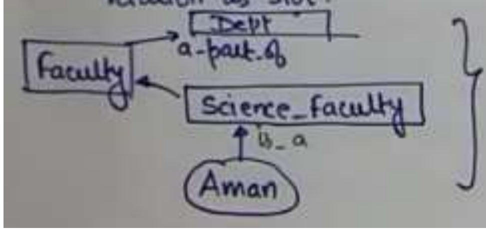
For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India

Representing Knowledge using FRAMES:-

FRAME is a Collection of attributes or Slots and associated Values that describe some real-world entity.

↳ uses datastructure (records) to represent the knowledge represented in semantic N/w.

↳ Each frame represents the node in semantic N/w as a class, or an instance and each relation as slot.



Attributes attached with each Slot:-

1. Instance: Relates Slot with class
2. Definition: slot defn / Value.
3. Default: Slot default Value.
4. Domain: Slot elements domain
5. Range: Specifies class of which elements.
6. Range Constrain: Logical expression] $6 < i < 10$
7. To-Compute: Value of Slot is to be computed.
8. Single-Valued: funcn returns Single Value.
9. Inverse: slot inverse Used in reasoning.
10. Transfers-Through: Inheritance.

Reasoning actions that can be performed Frames:-

① Relating the definition: isa, inverse links] Propagation of defⁿ to relate all infoⁿ.

② Inheritance: Inherited all values including default values of the slot.

③ Legality of Value: Checks the legality of slot value.] Range constraint.

④ Consistency check: Verifying slot value consistency before adding to the frame.] Domain

⑤ Maintaining Consistency: when one slot is updated its inverse should also be updated.

⑥ Computation of a Value of Slot: to compute transfers through.

SCRIPTS:- used for Knowledge Representation.

↳ Script is a structure that prescribes a set of circumstances which could be expected to follow on from one another.

↳ Considered to consist of a no. of slots or frames but with more specialized roles.

Components of a SCRIPT:-

① Roles:- Persons involved in Event.] Student.

② Props: objects involved in Event.] Pen, i_card, answer_sheet.

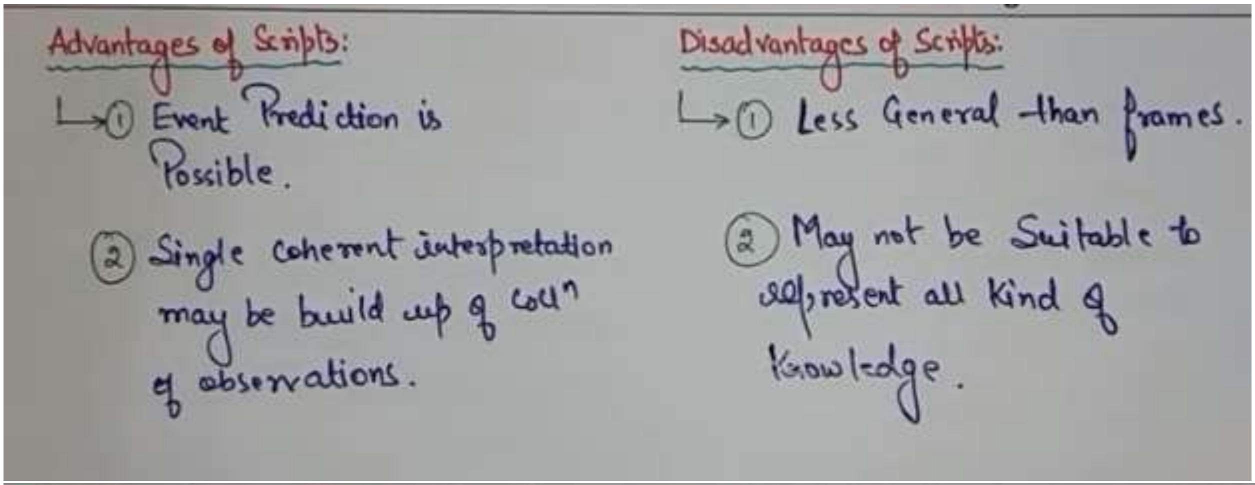
③ Entry Conditions: Conditions that needs to be satisfied before event occur in script] i_card must be present with the student.

④ Results: Condⁿ that will be true after event in script.] Filled_answer Sheet.

⑤ Track: Variations on the Script.] CAT, AIEEE } Exam_Centre

⑥ Scenes: Sequence of event that occurs.

exam hall entry
↓
Getting allocated seat
↓
Getting Question Paper
↳ Writing answers → Submit ans. sheet.



Advantages of Scripts:

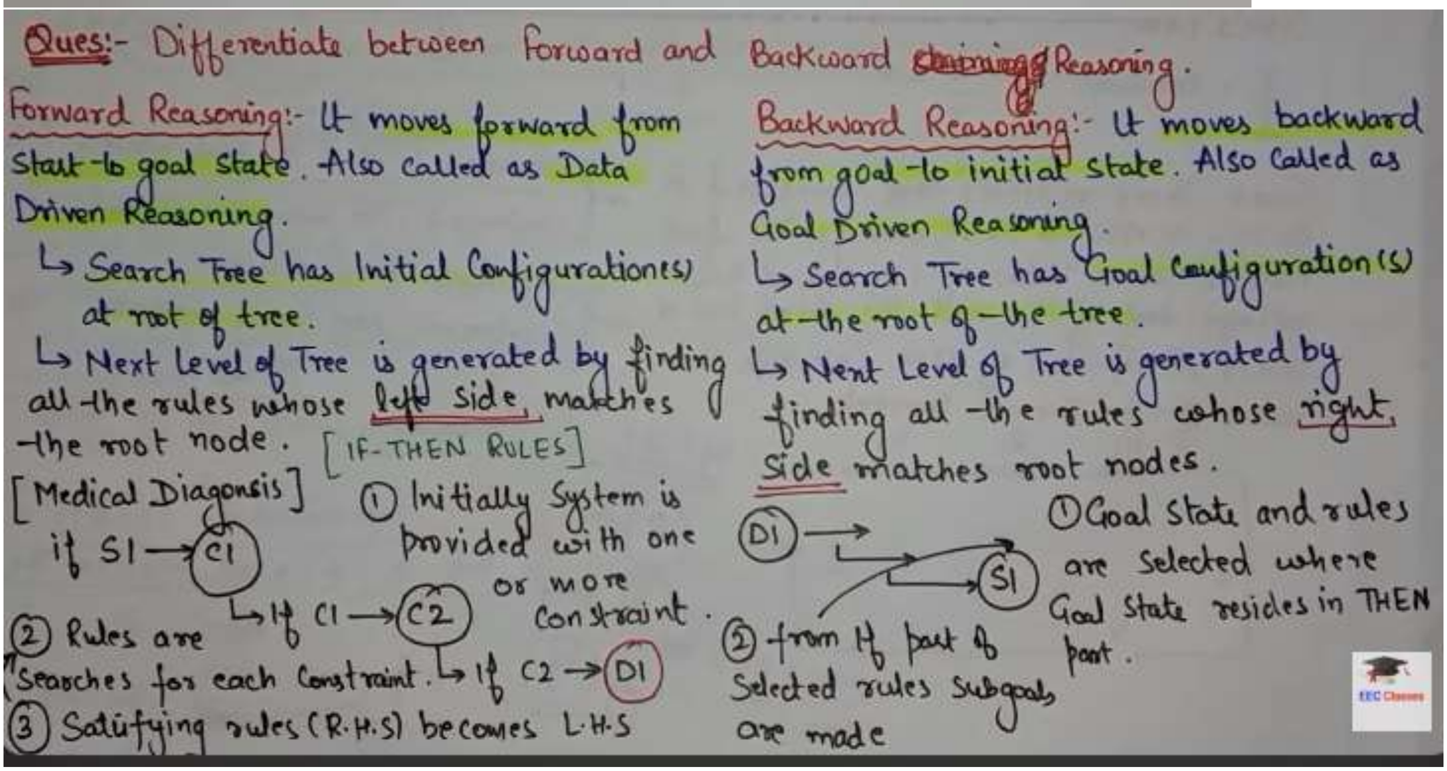
↳ ① Event Prediction is Possible.

② Single coherent interpretation may be build up of coll^n of observations.

Disadvantages of Scripts:

↳ ① Less General than frames.

② May not be Suitable to represent all kind of Knowledge.



Ques:- Differentiate between Forward and Backward ~~Chaining~~ Reasoning.

Forward Reasoning:- It moves forward from Start to goal state. Also called as Data Driven Reasoning.

↳ Search Tree has Initial Configuration(s) at root of tree.

↳ Next Level of Tree is generated by finding all the rules whose left side matches the root node. [IF-THEN RULES]

[Medical Diagonsis]

① Initially System is provided with one or more constraint.

if S1 → C1

↳ if C1 → C2

↳ if C2 → D1

② Rules are Searches for each Constraint.

③ Satisfying rules (R.H.S) becomes L.H.S

Backward Reasoning:- It moves backward from goal to initial state. Also Called as Goal Driven Reasoning.

↳ Search Tree has Goal Configuration(s) at the root of the tree.

↳ Next Level of Tree is generated by finding all the rules whose right Side matches root nodes.

D1 → S1

① Goal State and rules are Selected where Goal State resides in THEN part.

② from If part of Selected rules Subgoals are made

Hill Climbing Algorithm in Artificial Intelligence

- Hill climbing algorithm is a local search algorithm which continuously moves in the direction of increasing elevation/value to find the peak of the mountain or best solution to the problem. It terminates when it reaches a peak value where no neighbor has a higher value.
- Hill climbing algorithm is a technique which is used for optimizing the mathematical problems. One of the widely discussed examples of Hill climbing algorithm is Traveling-salesman Problem in which we need to minimize the distance traveled by the salesman.
- It is also called greedy local search as it only looks to its good immediate neighbor state and not beyond that.
- A node of hill climbing algorithm has two components which are state and value.

- Hill Climbing is mostly used when a good heuristic is available.
- In this algorithm, we don't need to maintain and handle the search tree or graph as it only keeps a single current state.

Features of Hill Climbing:

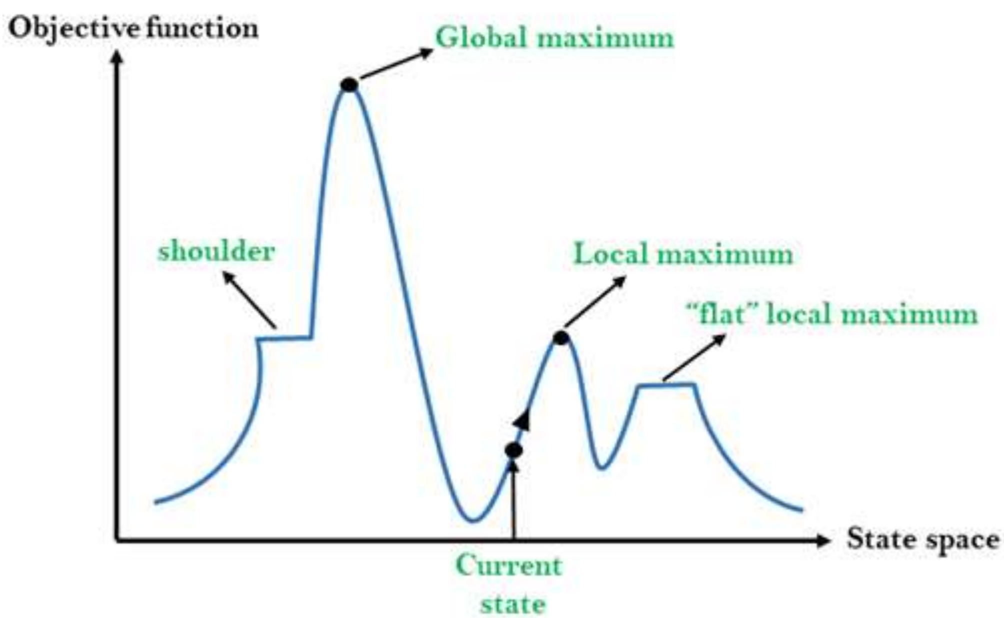
Following are some main features of Hill Climbing Algorithm:

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

State-space Diagram for Hill Climbing:

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

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



Different regions in the state space landscape:

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

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

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

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

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

Types of Hill Climbing Algorithm:

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

1. Simple Hill Climbing:

Simple hill climbing is the simplest way to implement a hill climbing algorithm. **It only evaluates the neighbor node state at a time and selects the first one which optimizes current cost and set it as a current state**. It only checks it's one successor state, and if it finds better than the current state, then move else be in the same state. This algorithm has the following features:

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

Algorithm for Simple Hill Climbing:

- **Step 1:** Evaluate the initial state, if it is goal state then return success and Stop.
- **Step 2:** Loop Until a solution is found or there is no new operator left to apply.
- **Step 3:** Select and apply an operator to the current state.
- **Step 4:** Check new state:
  a. If it is goal state, then return success and quit.
  b. Else if it is better than the current state then assign new state as a current state.
  c. 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.
  a. Let SUCC be a state such that any successor of the current state will be better than it.
  b. For each operator that applies to the current state:
     a. Apply the new operator and generate a new state.
     b. Evaluate the new state.
     c. If it is goal state, then return it and quit, else compare it to the SUCC.
     d. If it is better than SUCC, then set new state as SUCC.
     e. If the SUCC is better than the current state, then set current state to SUCC.
- **Step 5:** Exit.

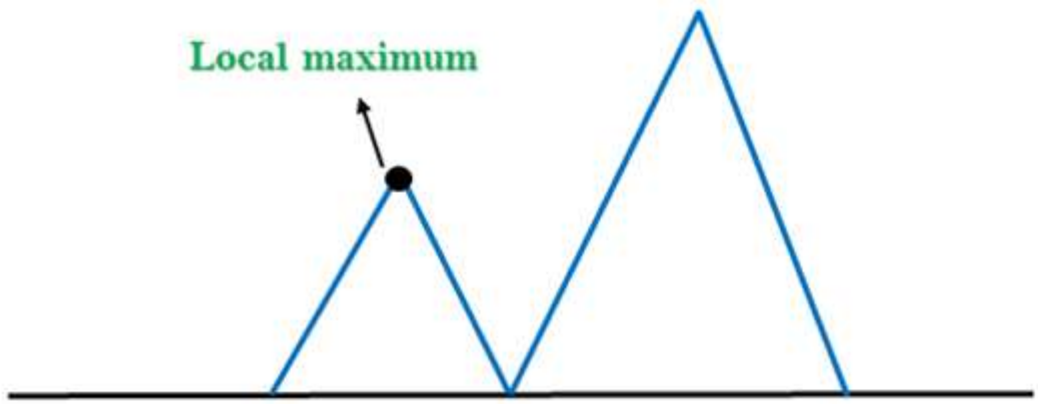
3. Stochastic hill climbing:

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:

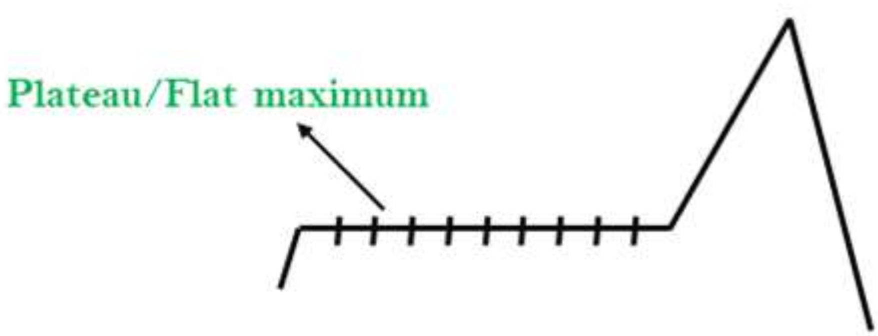
1. Local Maximum: A local maximum is a peak state in the landscape which is better than each of its neighboring states, but there is another state also present which is higher than the local maximum.

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.

Advantages of Hill Climbing algorithm:

1. Hill Climbing is a simple and intuitive algorithm that is easy to understand and implement.
2. It can be used in a wide variety of optimization problems, including those with a large search space and complex constraints.
3. Hill Climbing is often very efficient in finding local optima, making it a good choice for problems where a good solution is needed quickly.
4. The algorithm can be easily modified and extended to include additional heuristics or constraints.

Disadvantages of Hill Climbing algorithm:

1. Hill Climbing can get stuck in local optima, meaning that it may not find the global optimum of the problem.
2. The algorithm is sensitive to the choice of initial solution, and a poor initial solution may result in a poor final solution.
3. Hill Climbing does not explore the search space very thoroughly, which can limit its ability to find better solutions.
4. It may be less effective than other optimization algorithms, such as genetic algorithms or simulated annealing, for certain types of problems.

. Stochastic hill climbing:

- Evaluate the initial state. If it is a goal state then stop and return success. Otherwise, make the initial state the current state.
- Repeat these steps until a solution is found or the current state does not change.
  - Select a state that has not been yet applied to the current state.
  - Apply the successor function to the current state and generate all the neighbor states.

- Among the generated neighbor states which are better than the current state choose a state randomly (or based on some probability function).
- If the chosen state is the goal state, then return success, else make it the current state and repeat step 2 of the second point.

- Exit from the function.

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:

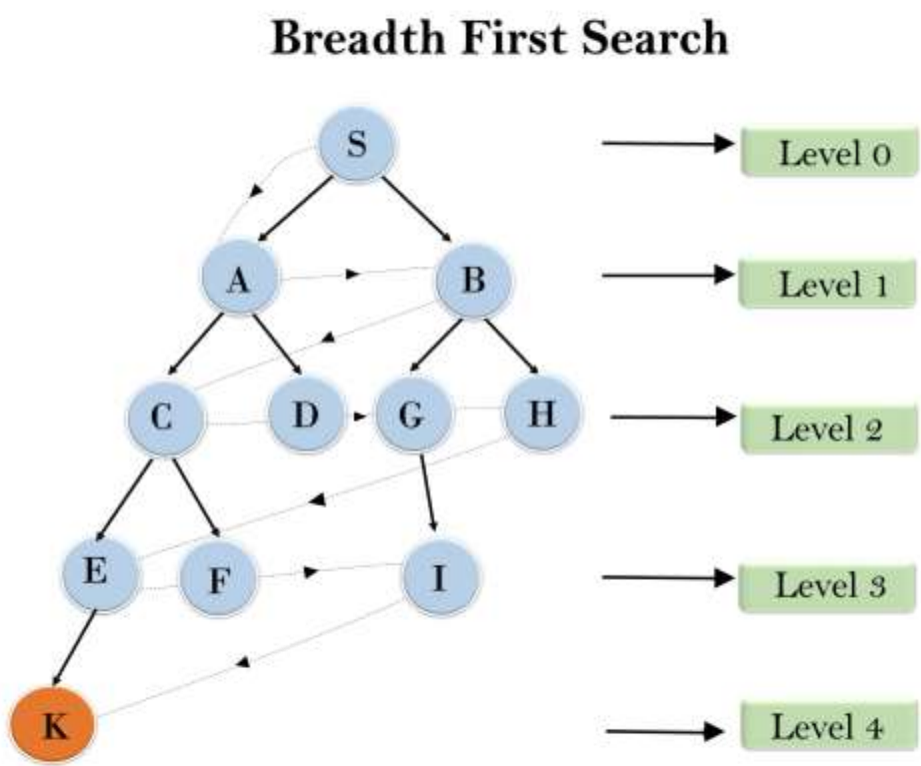
- It requires lots of memory since each level of the tree must be saved into memory to expand the next level.

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

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

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

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

2. Depth-first Search

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

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:

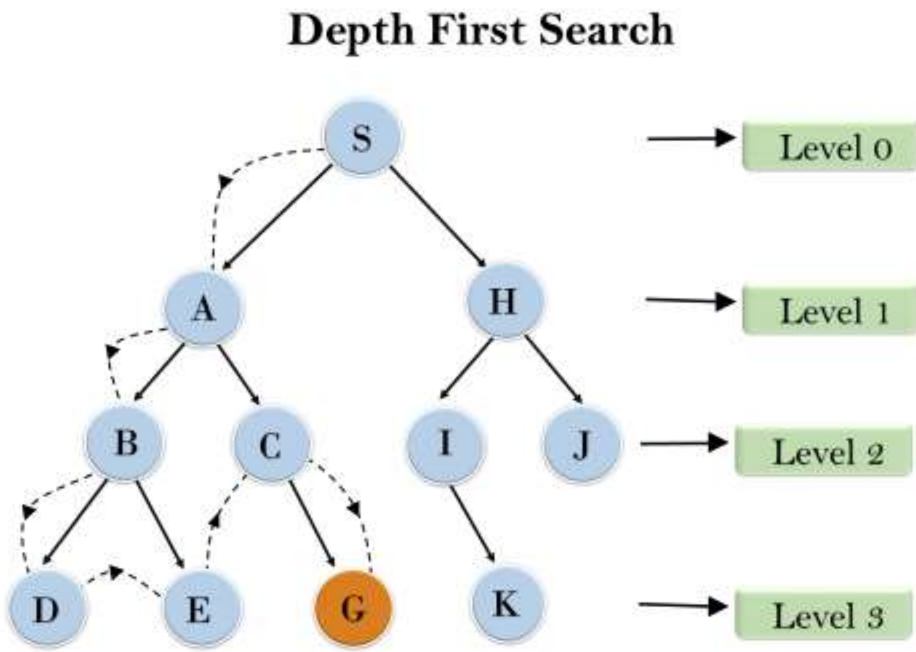
- There is the possibility that many states keep re-occurring, and there is no guarantee of finding the solution.
- DFS algorithm goes for deep down searching and sometime it may go to the infinite loop.

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:

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

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

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

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

3. Depth-Limited Search Algorithm:

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

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

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

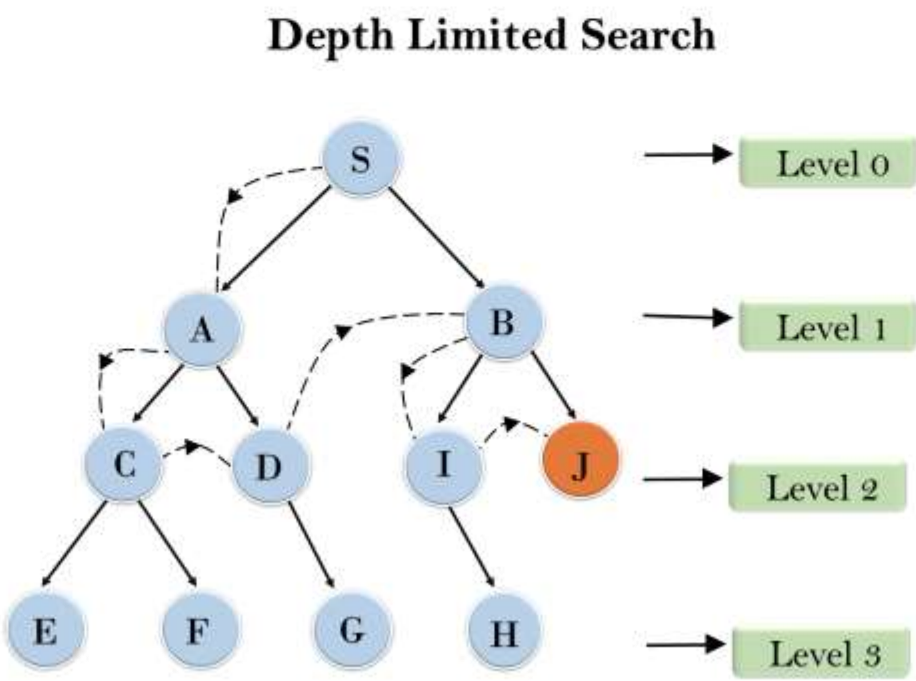
Advantages:

Depth-limited search is Memory efficient.

Disadvantages:

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

Example:



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

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

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

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

4. 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.

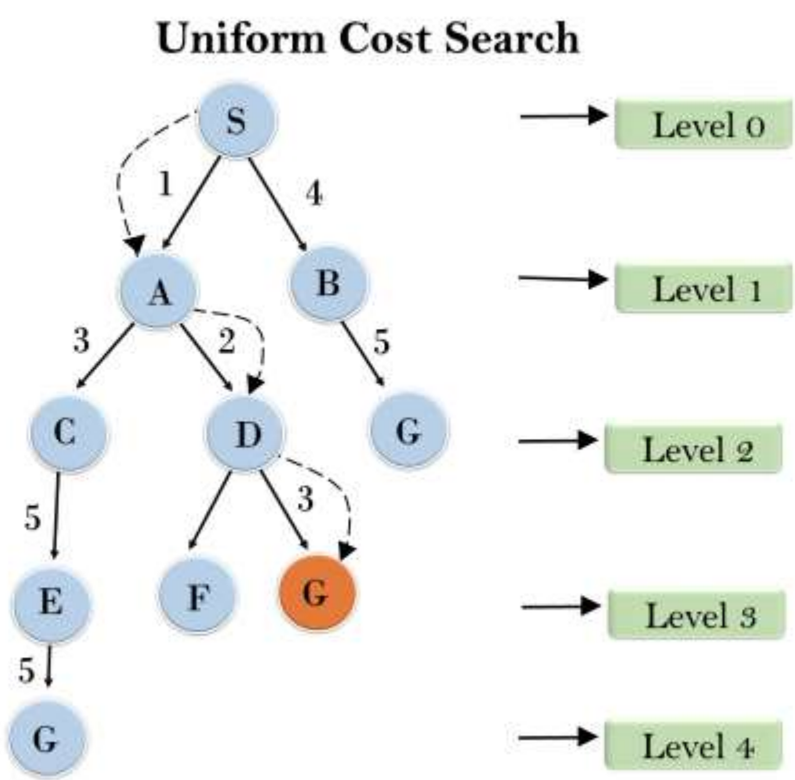
Advantages:

- 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**$O(b^{1 + [C^*/\varepsilon]})$/**.

Space Complexity:

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

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:

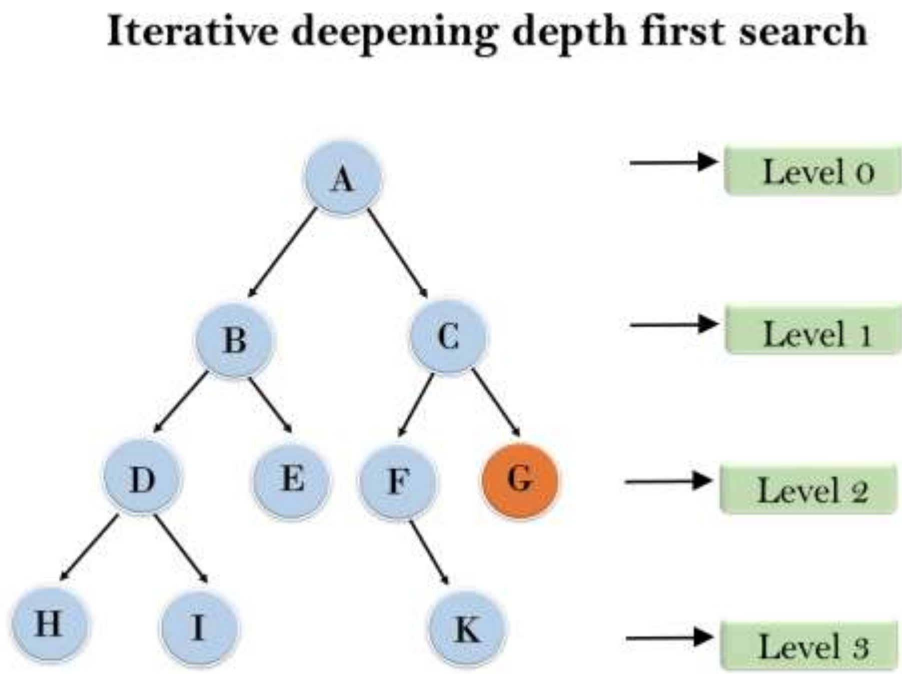
- Itcombines the benefits of BFS and DFS search algorithm in terms of fast search and memory efficiency.

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 $\mathbf{O(b^d)}$.

Space Complexity:

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

Optimal:

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

6. Bidirectional Search Algorithm:

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

Bidirectional search replaces one single search graph with two small subgraphs in which one starts the search from an initial vertex and other starts from goal vertex. The search stops when these two graphs intersect each other.

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

Advantages:

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

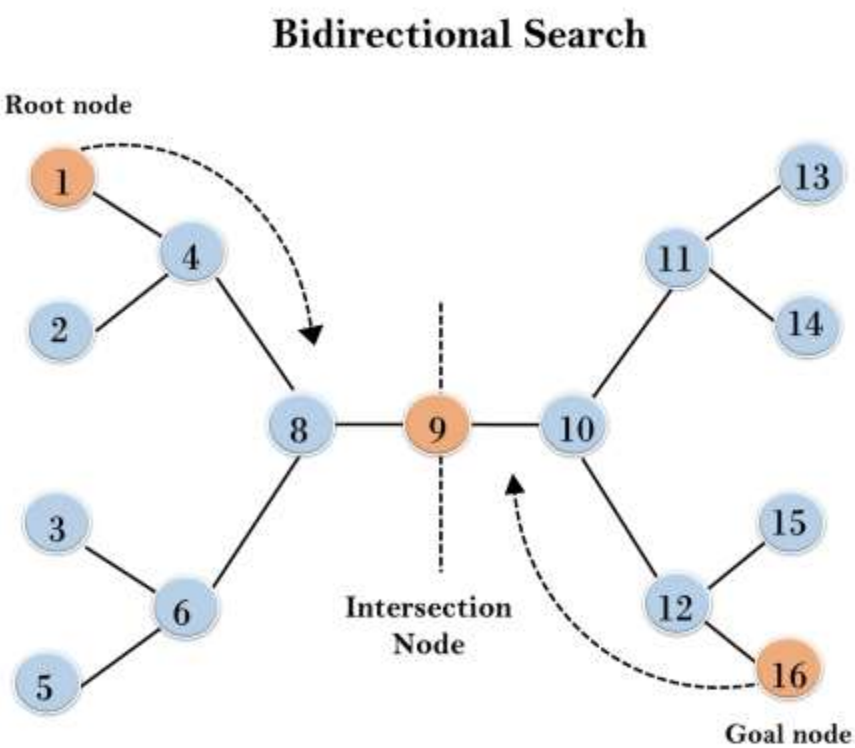
Disadvantages:

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

Example:

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

The algorithm terminates at node 9 where two searches meet.



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

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

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

Optimal: Bidirectional search is Optimal.

Informed Search Algorithms

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.

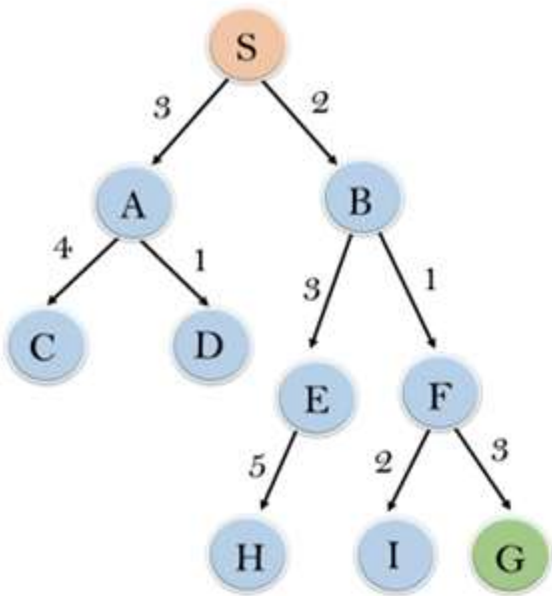
Disadvantages:

- 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.

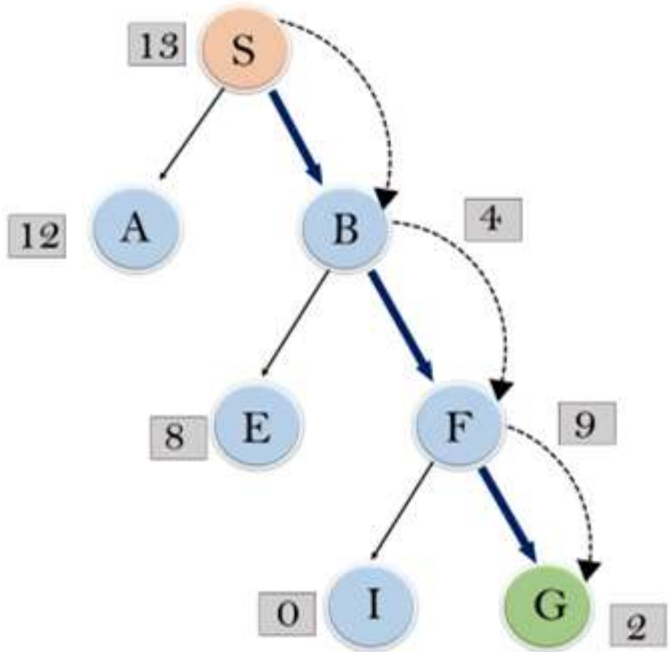
Example:

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.



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

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



Expand the nodes of S and put in the CLOSED list

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

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

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

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

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

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

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

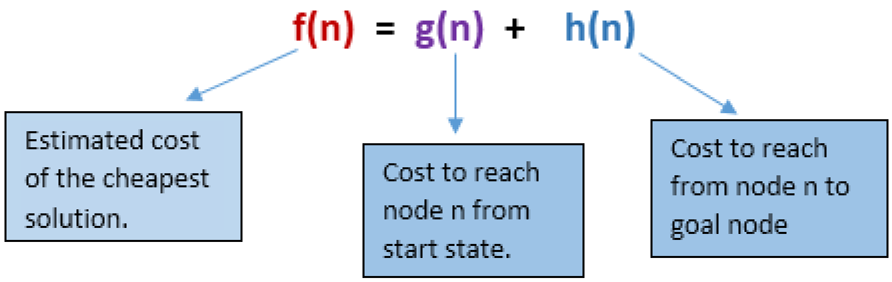
Complete: Greedy best-first search is also incomplete, even if the given state space is finite.

Optimal: Greedy best first search algorithm is not optimal.

2.) A* Search Algorithm:

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

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



At each point in the search space, only those 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.

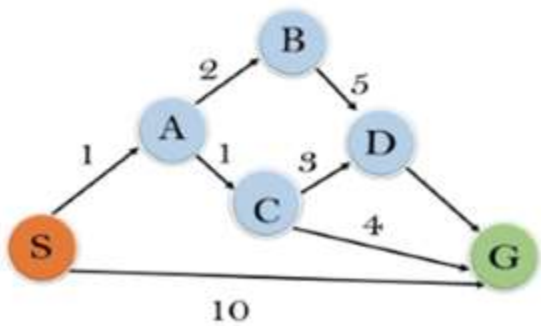
Disadvantages:

- 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.

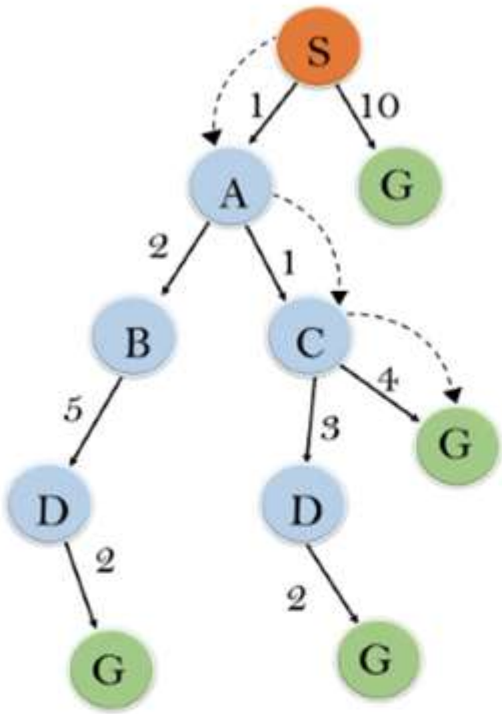
Example:

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.



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

Solution:



Initialization: {(S, 5)}

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

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

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

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

Points to remember:

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

Complete: A* algorithm is complete as long as:

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

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

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

- **Consistency:** Second required condition is consistency for only A* graph-search.

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

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

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

Means-Ends Analysis in Artificial Intelligence

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

How means-ends analysis Works:

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

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

Operator Subgoaling

In the MEA process, we detect the differences between the current state and goal state. Once these differences occur, then we can apply an operator to reduce the differences. But sometimes it is possible that an operator cannot be applied to the current state. So we create the subproblem of the current state, in which operator can be applied, such type of backward chaining in which operators are selected, and then sub goals are set up to establish the preconditions of the operator is called **Operator Subgoaling**.

Algorithm for Means-Ends Analysis:

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

- **Step 1:** Compare CURRENT to GOAL, if there are no differences between both then return Success and Exit.
- **Step 2:** Else, select the most significant difference and reduce it by doing the following steps until the success or failure occurs.

a. Select a new operator O which is applicable for the current difference, and if there is no such operator, then signal failure.

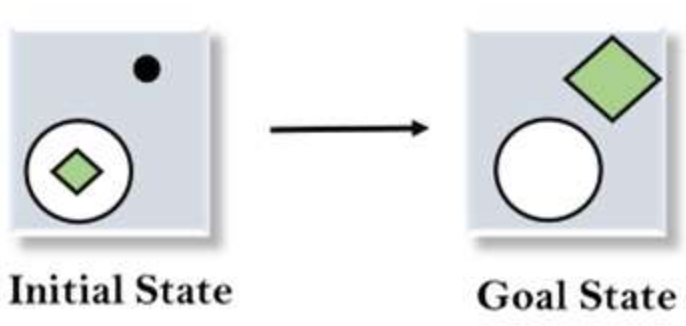
   b. Attempt to apply operator O to CURRENT. Make a description of two states.
   i) O-Start, a state in which O?s preconditions are satisfied.
   ii) O-Result, the state that would result if O were applied In O-start.

   c. 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.

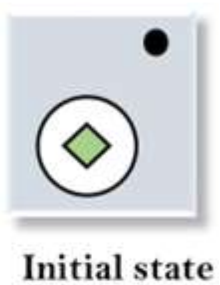


Solution:

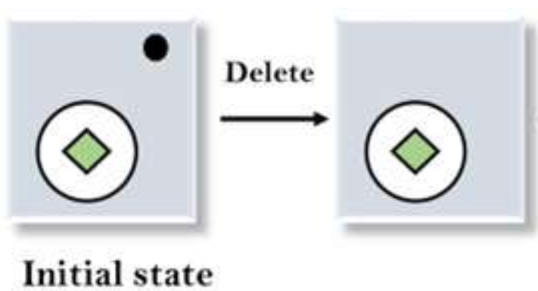
To solve the above problem, we will first find the differences between initial states and goal states, and for each difference, we will generate a new state and will apply the operators. The operators we have for this problem are:

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

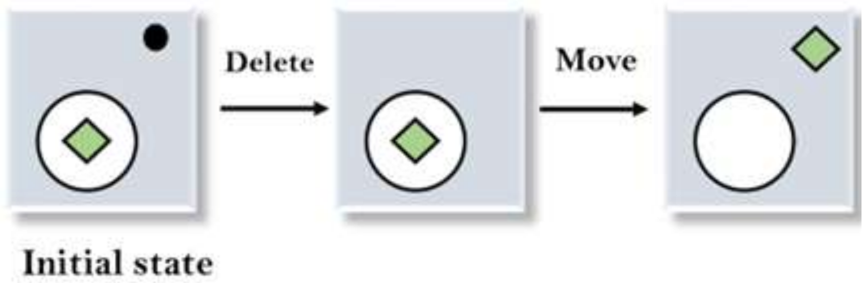
1. Evaluating the initial state: In the first step, we will evaluate the initial state and will compare the initial and Goal state to find the differences between both states.



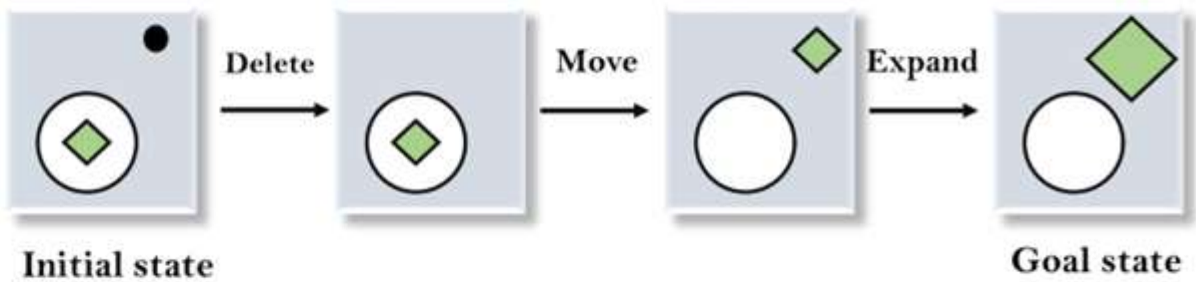
2. Applying Delete operator: As we can check the first difference is that in goal state there is no dot symbol which is present in the initial state, so, first we will apply the **Delete operator** to remove this dot.



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.



est First Search (Informed Search)

-

In BFS and DFS, when we are at a node, we can consider any of the adjacent as the next node. So both BFS and DFS blindly explore paths without considering any cost function.

*The idea of **Best First Search** is to use an evaluation function to decide which adjacent is most promising and then explore.*

Best First Search falls under the category of Heuristic Search or Informed Search.

Implementation of Best First Search:

We use a priority queue or heap to store the costs of nodes that have the lowest evaluation function value. So the implementation is a variation of BFS, we just need to change Queue to PriorityQueue.

// Pseudocode for Best First Search

Best-First-Search(Graph g, Node start)

1) Create an empty PriorityQueue

PriorityQueue **pq**;

2) Insert "start" in pq.

pq.insert(start)

3) Until PriorityQueue is empty

u = PriorityQueue.DeleteMin

If u is the goal

Exit

Else

Foreach neighbor v of u

If v "Unvisited"

Mark v "Visited"

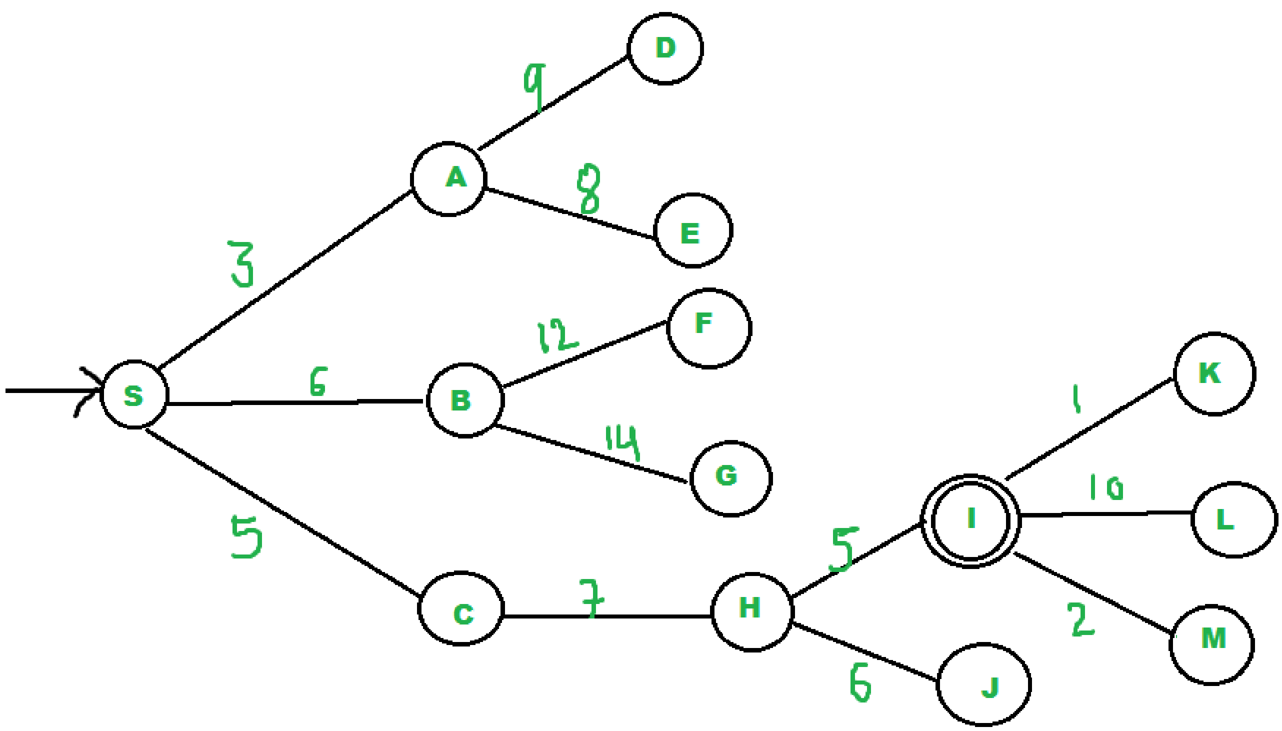
pq.insert(v)

Mark u "Examined"

End procedure

Illustration:

Let us consider the below example:



- *We start from source "S" and search for goal "I" using given costs and Best First search.*

- *pq initially contains S*

  - *We remove S from pq and process unvisited neighbors of S to pq.*
  - *pq now contains {A, C, B} (C is put before B because C has lesser cost)*

- *We remove A from pq and process unvisited neighbors of A to pq.*
  - *pq now contains {C, B, E, D}*

- *We remove C from pq and process unvisited neighbors of C to pq.*
  - *pq now contains {B, H, E, D}*

- *We remove B from pq and process unvisited neighbors of B to pq.*
  - *pq now contains {H, E, D, F, G}*
- *We remove H from pq.*
- *Since our goal "I" is a neighbor of H, we return*

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:

```
function minimax(node, depth, maximizingPlayer) is
if depth ==0 or node is a terminal node then
```

```
return static evaluation of node

if MaximizingPlayer then      // for Maximizer Player
maxEva= -infinity
 for each child of node do
 eva= minimax(child, depth-1, false)
maxEva= max(maxEva,eva)       //gives Maximum of the values
return maxEva

else                         // for Minimizer player
 minEva= +infinity
 for each child of node do
 eva= minimax(child, depth-1, true)
 minEva= min(minEva, eva)       //gives minimum of the values
 return minEva
```

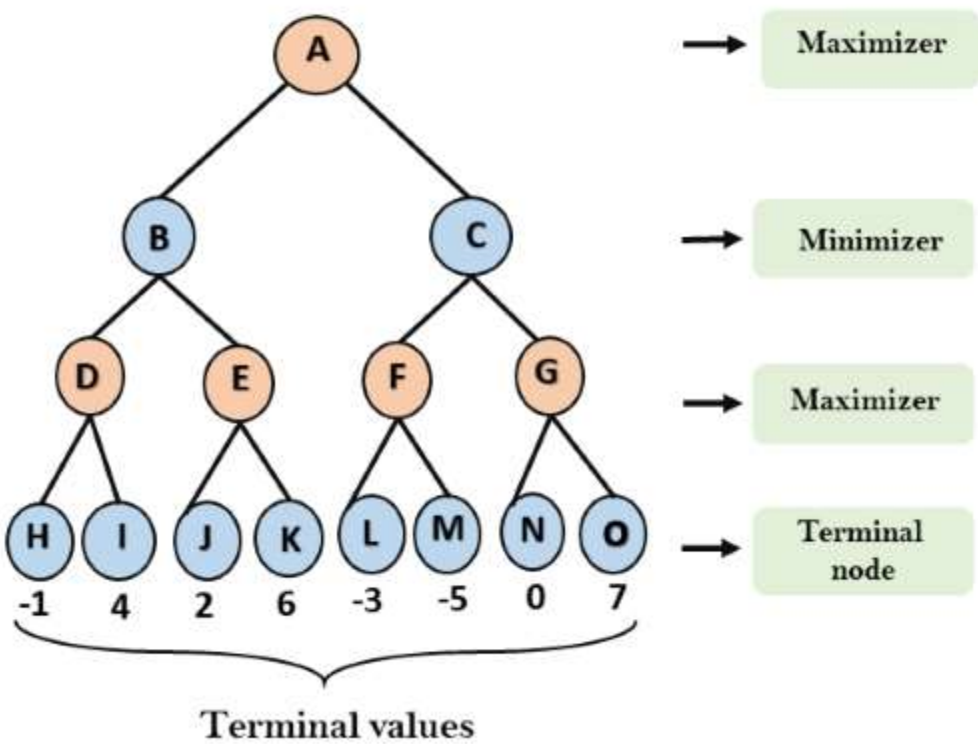
Initial call:

Minimax(node, 3, true)

Working of Min-Max Algorithm:

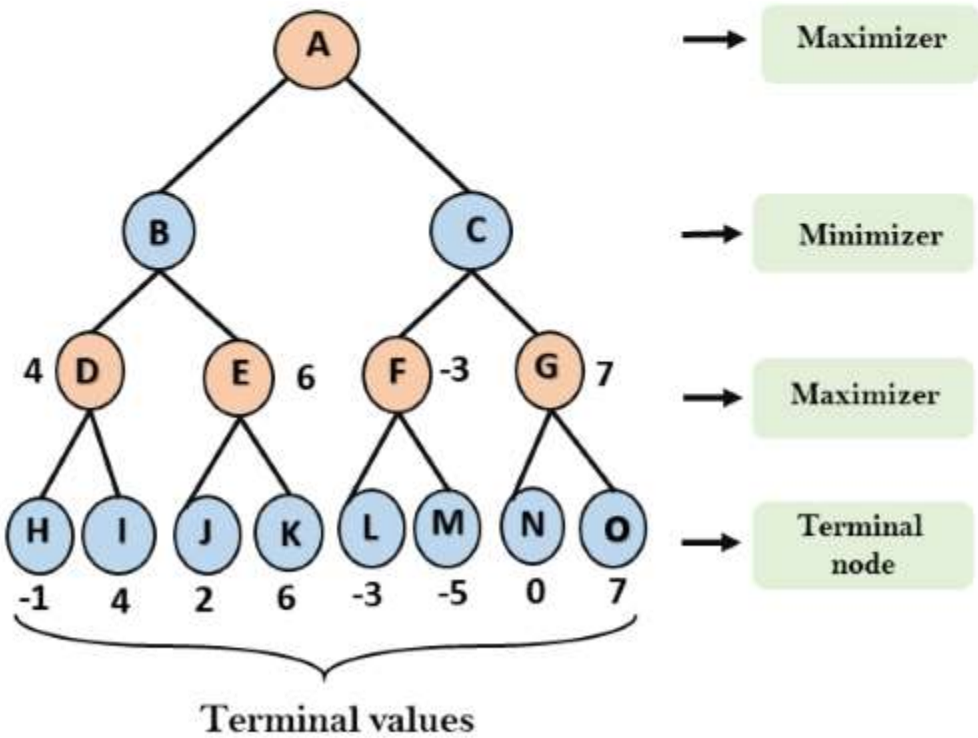
- The working of the minimax algorithm can be easily described using an example. Below we have taken an example of game-tree which is representing the two-player game.
- In this example, there are two players one is called Maximizer and other is called Minimizer.
- Maximizer will try to get the Maximum 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.



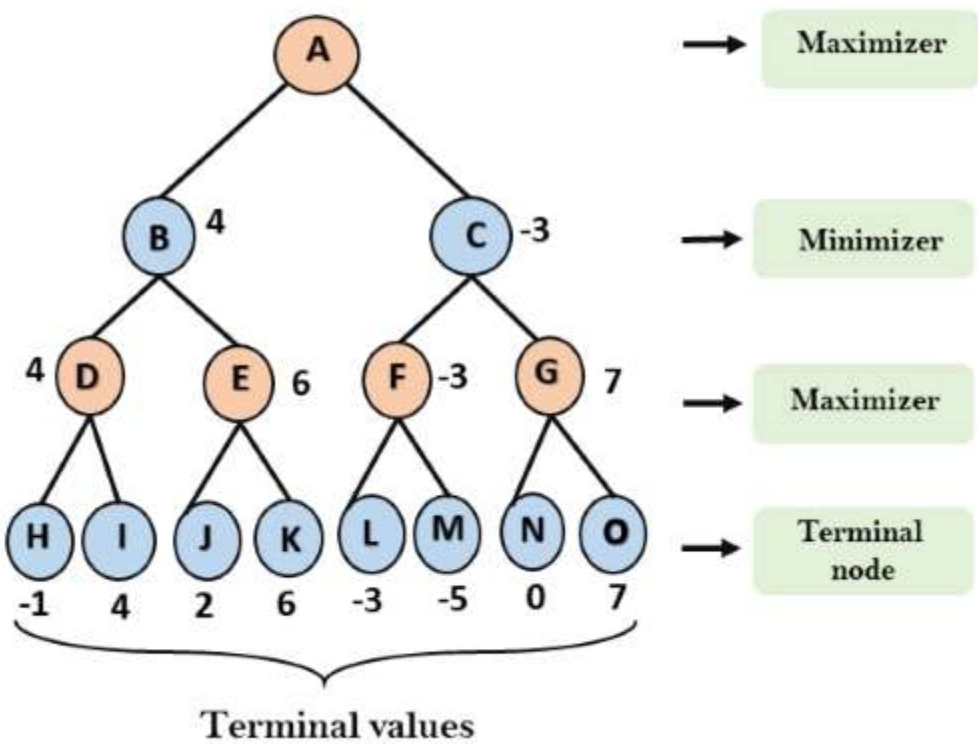
Step 2: Now, first we find the utilities value for the Maximizer, its initial value is -∞, so we will compare each value in terminal state with initial value of Maximizer and determines the higher nodes values. It will find the maximum among the all.

- 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



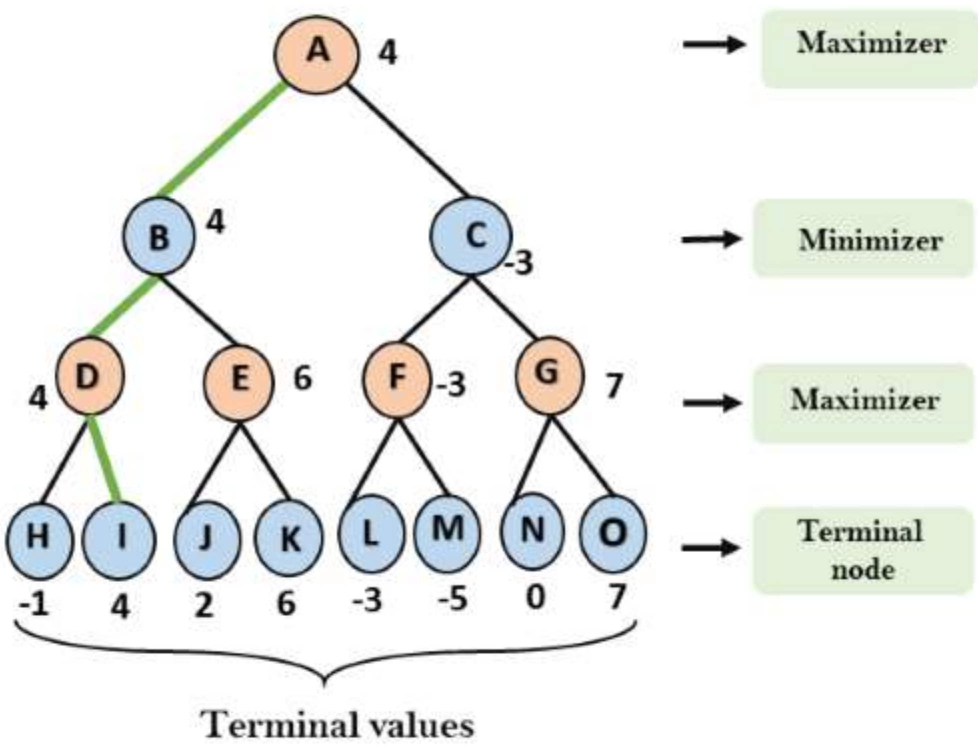
Step 3: In the next step, it's a turn for minimizer, so it will compare all nodes value with +∞, and will find the 3rd layer node values.

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



Step 4: 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 **$O(b^m)$**, 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.

Intelligent agents are entities designed to perceive their environment and take actions to maximize their chances of successfully achieving their goals. They can be software-based, hardware-based, or a combination of both, and their nature can be characterized by several key attributes:

1. **Autonomy:** Intelligent agents operate independently, making decisions and taking actions without direct human intervention. They have a degree of autonomy in deciding how to achieve their objectives.
2. **Perception:** Agents gather information from their environment through sensors or by receiving input. Perception allows them to understand the state of the world and make informed decisions.
3. **Reasoning and Decision-Making:** Based on the information perceived, agents use reasoning and decision-making processes to analyze situations and choose actions that are likely to lead to successful outcomes.
4. **Learning:** Intelligent agents can adapt and improve their performance over time through learning. They might learn from experience, feedback, or by analyzing patterns in data.
5. **Goal-Oriented Behavior:** Agents are designed to pursue specific goals or objectives. They take actions that aim to achieve these goals, adjusting their strategies based on changing circumstances.
6. **Interaction:** Agents often interact with other agents or entities in their environment. This interaction might involve cooperation, competition, negotiation, or coordination to achieve their objectives.
7. **Adaptability:** Intelligent agents can adapt to changes in their environment or goals. They might adjust their behavior or strategies in response to new information or unexpected situations.

You

what is belief and plausabilty in dempster thoery

ChatGPT

ChatGPT

In Dempster-Shafer Theory, belief and plausibility are measures used to quantify uncertainty or degrees of belief about the occurrence of events. These measures play a crucial role in reasoning under uncertainty and are fundamental concepts in the theory of evidence.

1. **Belief:** Belief represents the lower bound of the probability of an event or hypothesis based on available evidence. It measures the extent to which the evidence supports the event, considering all the relevant information. Mathematically, belief in an event A is denoted by the function Bel(A). It accounts for the evidence supporting A and all its subsets.
2. **Plausibility:** Plausibility, on the other hand, represents the upper bound of the probability of an event or hypothesis. It measures how plausible or reasonable the event is given the available evidence. Mathematically, plausibility in an event A is denoted by the function Pl(A). It accounts for the evidence supporting A or any superset of A.

In simpler terms, belief assigns a value to how much evidence supports a particular event, while plausibility measures the extent to which the evidence does not contradict the event.

"Generate and Test" and "Hill Climbing" are both problem-solving strategies used in artificial intelligence, but they operate differently in the search for solutions. Here's a comparison between the two with examples:

Generate and Test:

- **Approach:** In this strategy, a large set of potential solutions is generated, and each solution is tested against a predefined criterion to determine if it satisfies the problem requirements.
- **Process:** It involves systematically generating potential solutions, typically through brute-force or exhaustive methods, and then evaluating each solution against specific criteria.
- **Example:** Consider finding a password that opens a lock. Using a "generate and test" approach, one might generate various combinations of numbers or characters and test each combination to see if it unlocks the system.

Hill Climbing:

- **Approach:** Hill climbing is a local search algorithm that starts with an initial solution and iteratively moves towards the best solution by making incremental improvements at each step.
- **Process:** It evaluates neighboring solutions and selects the one that improves the current solution the most, moving in the direction that leads to an increase in the evaluation metric (e.g., minimizing or maximizing a function).
- **Example:** Imagine finding the highest point on a terrain using hill climbing. Starting from a random location, you move to neighboring locations with higher elevation until you reach a peak. However, hill climbing can get stuck in local optima, which might not be the global optimum (highest point).

Comparison:

- **Search Space:** Generate and test examines a broader search space by generating multiple potential solutions, while hill climbing explores a smaller, localized search space around the initial solution.
- **Exploration vs. Exploitation:** Generate and test involves more exploration of the solution space, whereas hill climbing focuses on exploiting the neighborhood of the current solution for incremental improvements.
- **Optimality:** Generate and test may find an optimal solution but might be computationally expensive for larger search spaces. Hill climbing tends to find local optima and might miss the global optimum in some cases.

Both strategies have their strengths and weaknesses, and their effectiveness depends on the nature of the problem, the size of the search space, and the specific requirements for finding a solution.

Aspect	Generate and Test	Hill Climbing
Approach	Generates a set of potential solutions and tests each against criteria	Starts with an initial solution and iteratively moves towards better solutions
Search Space	Explores a broader solution space	Explores a localized space around the initial solution
Exploration vs. Exploitation	Emphasizes exploration of solutions	Focuses on exploiting the neighborhood for incremental improvements
Optimality	May find an optimal solution but can be computationally expensive	Finds local optima; might not reach the global optimum
Process	Involves generating many solutions and testing each separately	Involves iteratively improving solutions by making incremental changes
Example	Searching for a password by trying various combinations	Finding the highest point on a terrain by climbing towards higher elevations

Mean-End Analysis (MEA):

- **Approach:** MEA is a problem-solving technique that works backward from the goal state by analyzing differences between the current state and the goal state. It focuses on reducing the difference between the current and the desired state in each step.
- **Process:** It decomposes the problem into subgoals and identifies the actions needed to achieve each subgoal, gradually moving closer to the final solution.
- **Example:** Consider solving a maze using MEA. You start from the exit (goal state) and work backward, identifying the steps needed to reach the exit from your current position. At each step, you analyze the differences between your current location and the exit, determining the actions required to reduce this difference until you reach the starting point.

Minimax Algorithm:

- **Approach:** Minimax is used in decision-making and game theory, especially in zero-sum games, to minimize the potential loss while maximizing the potential gain.
- **Process:** It explores a game tree by considering all possible moves and opponent responses up to a certain depth, assigning values to each move based on the assumption that the opponent will make the best moves for themselves. It aims to choose the move that maximizes the minimum possible outcome.

- **Example:** In chess, the minimax algorithm is used to evaluate potential moves. When considering a move, the algorithm looks ahead to possible opponent moves and assesses the best move to make, assuming the opponent will also make optimal moves to minimize the player's advantage.

Comparison:

Aspect	Mean-End Analysis	Minimax Algorithm
Purpose	Problem-solving strategy for achieving goals	Decision-making strategy, often in game theory
Application	Problem-solving in various domains	Game-playing scenarios, especially zero-sum games
Focus	Reducing differences between current and goal states	Minimizing potential loss while maximizing gain based on opponent moves
Usage	Problem-solving by decomposing into subgoals	Decision-making by exploring possible game outcomes
Examples	Solving mazes, planning algorithms	Game-playing AI in chess, tic-tac-toe, etc.

While both MEA and Minimax have different applications and approaches, they aim to systematically reach favorable outcomes, whether in problem-solving or strategic decision-making, by analyzing potential states and actions.

approaches to knowledge representations

ChatGPT

ChatGPT

Sure, knowledge representation refers to the way information is structured and organized in artificial intelligence systems. There are several approaches to representing knowledge:

1. Semantic Networks:

- Represent knowledge using nodes (concepts) connected by edges (relationships).
- Example: Representing relationships between different animal species (nodes) based on their characteristics (edges).

2. Frames:

- Represent knowledge as a collection of attributes and values organized in a hierarchical structure.
- Example: Describing a car as a frame with attributes like make, model, color, and engine type.

3. Production Rules:

- Represent knowledge in a set of conditional statements (IF-THEN rules) to guide decision-making.

- Example: Rules for diagnosing diseases based on symptoms.

4. Predicate Logic:

- Represent knowledge using logical symbols and quantifiers to express relationships and constraints.
- Example: Representing "All humans are mortal" as ∀x (Human(x) → Mortal(x)).

5. Ontologies:

- Organize knowledge in a formal, explicit way by defining concepts, entities, and their relationships within a specific domain.
- Example: Developing an ontology for the domain of medicine to represent relationships between diseases, symptoms, and treatments.

6. Bayesian Networks:

- Represent probabilistic relationships among variables using a graphical model.
- Example: Representing dependencies between symptoms and diseases in medical diagnosis.

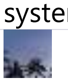
7. Neural Networks:

- Represent knowledge through interconnected nodes in layers, learning patterns from data.
- Example: Training a neural network to recognize images by learning features from labeled data.

8. Description Logics:

- Represent knowledge using a formal logic to describe the semantics of concepts and relationships.
- Example: Ontologies in the Semantic Web are often represented using description logics for precise knowledge representation.

Each approach to knowledge representation has its strengths and weaknesses, and the choice often depends on the specific problem domain, the nature of the knowledge to be represented, and the intended application within AI systems.



You

various types of knowledge

ChatGPT

ChatGPT

Knowledge can be categorized into various types based on its nature, source, or usage in different contexts within artificial intelligence and cognitive sciences. Here are some classifications:

1. Explicit Knowledge:

- **Declarative Knowledge:** Factual information, such as facts, concepts, principles, or rules, that can be articulated and documented.

2. Tacit Knowledge:

- **Procedural Knowledge:** Practical, skill-based knowledge gained through experience, often challenging to express explicitly.

3. Domain-Specific Knowledge:

- **Structured Knowledge:** Information specific to a particular domain or field, organized in a structured format.

4. Meta Knowledge:

- **Knowledge about Knowledge:** Understanding or knowledge about how information is created, processed, or used.

5. Situational Knowledge:

- **Contextual Knowledge:** Information relevant to a specific context or situation.

6. Common Sense Knowledge:

- **Implicit Knowledge:** General knowledge and understanding about everyday phenomena and human behavior.

7. Heuristic Knowledge:

- **Rule-of-Thumb Knowledge:** Informal strategies or guidelines used to solve problems or make decisions.

8. Tacit Knowledge:

- **Experiential Knowledge:** Knowledge gained through personal experience and practice.

9. Explicit Knowledge:

- **Documented Knowledge:** Knowledge that is recorded, documented, and easily transferable.

10. Cultural Knowledge:

- **Socially Constructed Knowledge:** Knowledge shaped by cultural practices, beliefs, norms, and traditions.

11. Meta-Knowledge:

- **Self-Awareness:** Knowledge about one's cognitive processes, abilities, and limitations.

12. Transactive Knowledge:

- **Shared Knowledge:** Knowledge shared and collectively developed within a group or community.

explain miniskey frames ...what are adv disadv

ChatGPT

ChatGPT

Minsky Frames, proposed by Marvin Minsky, are a knowledge representation technique used in artificial intelligence. They're a conceptual framework for organizing information and representing knowledge.

Advantages of Minsky Frames:

1. **Hierarchical Structure:** Frames allow the representation of knowledge in a hierarchical structure, making it easier to organize complex information into smaller, manageable units.
2. **Flexibility:** They permit flexibility in representing diverse types of information by allowing slots to contain various types of data.
3. **Modularity:** Frames are modular, allowing the creation of a template-like structure for different concepts. This modularity enables easy reuse and modification.
4. **Inheritance:** Inherited properties in frames facilitate the propagation of common characteristics from higher-level frames to lower-level ones, reducing redundancy.

Disadvantages of Minsky Frames:

1. **Complexity:** As systems grow, maintaining a large number of frames becomes complex and may lead to issues in managing extensive knowledge bases.
2. **Rigidity:** Frames may struggle to represent dynamic or evolving information, requiring constant updates or revisions as the domain changes.
3. **Semantic Ambiguity:** They might not always capture the nuances or complexities of real-world concepts, leading to ambiguity in representation.
4. **Efficiency:** Retrieval and inference in systems using frames might become computationally expensive with a large number of interconnected frames.

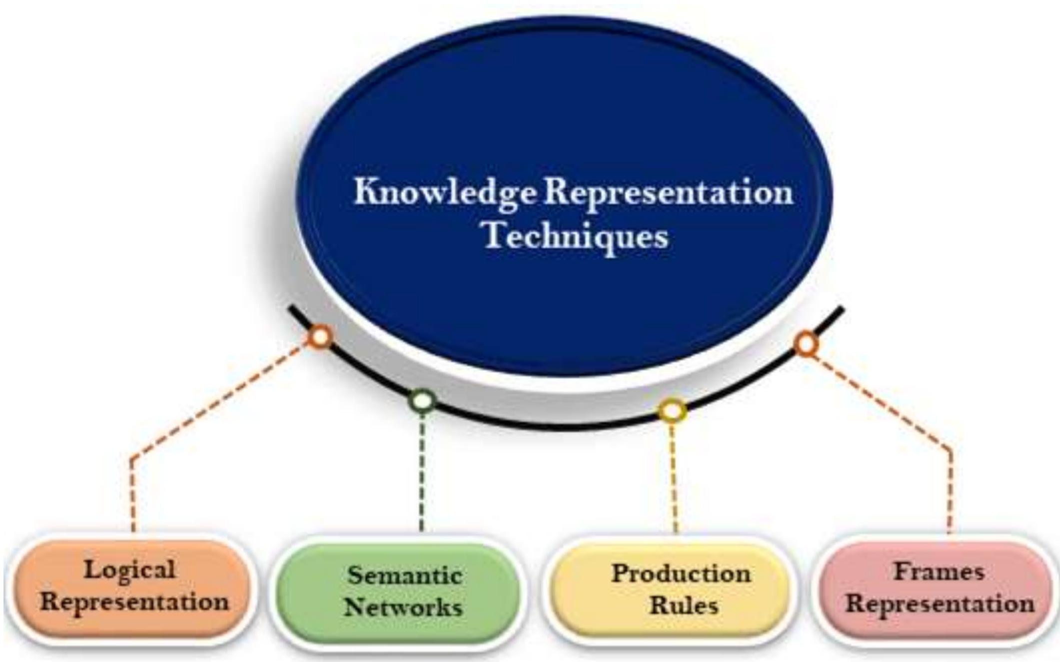
Minsky Frames have been influential in knowledge representation, particularly in early AI systems. While they offer a structured way to represent knowledge, they also have limitations in handling dynamic, evolving, or highly context-dependent information. Modern AI systems often combine various knowledge representation approaches to overcome these limitations.

Aspect	Minimax Algorithm	Alpha-Beta Pruning
Approach	Decision-making based on maximum minimum outcomes	Optimization technique in game tree traversal
Operation	Explores entire game tree, evaluating all moves	Traverses the tree while pruning irrelevant branches
Efficiency	Can be computationally expensive for deep trees	Significantly reduces nodes explored, more efficient
Evaluation	Aims to maximize minimum outcome	Same objective as Minimax but with pruning
Application	Used in game-playing scenarios	An optimization of Minimax in game-playing scenarios

Techniques of knowledge representation

There are mainly four ways of knowledge representation which are given as follows:

1. Logical Representation
2. Semantic Network Representation
3. Frame Representation
4. Production Rules



1. Logical Representation

Logical representation is a language with some concrete rules which deals with propositions and has no ambiguity in representation. Logical representation means drawing a conclusion based on various conditions. This representation lays down some important communication rules. It consists of precisely defined syntax and semantics which supports the sound inference. Each sentence can be translated into logics using syntax and semantics.

Syntax:

- Syntaxes are the rules which decide how we can construct legal sentences in the logic.
- It determines which symbol we can use in knowledge representation.
- How to write those symbols.

Semantics:

- Semantics are the rules by which we can interpret the sentence in the logic.
- Semantic also involves assigning a meaning to each sentence.

Logical representation can be categorised into mainly two logics:

a. Propositional Logics
b. Predicate logics

> Note: We will discuss Prepositional Logics and Predicate logics in later chapters.

Advantages of logical representation:

1. Logical representation enables us to do logical reasoning.
2. Logical representation is the basis for the programming languages.

Disadvantages of logical Representation:

1. Logical representations have some restrictions and are challenging to work with.
2. Logical representation technique may not be very natural, and inference may not be so efficient.

> Note: Do not be confused with logical representation and logical reasoning as logical representation is a representation language and reasoning is a process of thinking logically.

2. Semantic Network Representation

Semantic networks are alternative of predicate logic for knowledge representation. In Semantic networks, we can represent our knowledge in the form of graphical networks. This network consists of nodes representing objects and arcs which describe the relationship between those objects. Semantic networks can categorize the object in different forms and can also link those objects. Semantic networks are easy to understand and can be easily extended.

This representation consist of mainly two types of relations:

a. IS-A relation (Inheritance)

b. Kind-of-relation

Example: Following are some statements which we need to represent in the form of nodes and arcs.

Statements:

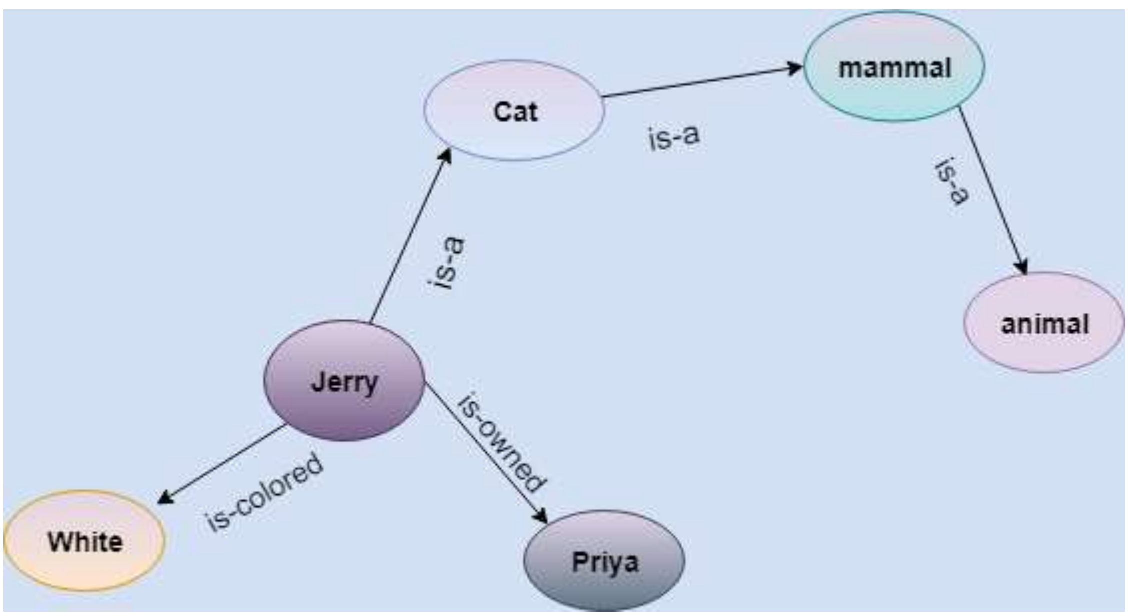
a. Jerry is a cat.

b. Jerry is a mammal

c. Jerry is owned by Priya.

d. Jerry is brown colored.

e. All Mammals are animal.



In the above diagram, we have represented the different type of knowledge in the form of nodes and arcs. Each object is connected with another object by some relation.

Drawbacks in Semantic representation:

1. Semantic networks take more computational time at runtime as we need to traverse the complete network tree to answer some questions. It might be possible in the worst case scenario that after traversing the entire tree, we find that the solution does not exist in this network.

2. Semantic networks try to model human-like memory (Which has 1015 neurons and links) to store the information, but in practice, it is not possible to build such a vast semantic network.
3. These types of representations are inadequate as they do not have any equivalent quantifier, e.g., for all, for some, none, etc.
4. Semantic networks do not have any standard definition for the link names.
5. These networks are not intelligent and depend on the creator of the system.

Advantages of Semantic network:

1. Semantic networks are a natural representation of knowledge.
2. Semantic networks convey meaning in a transparent manner.
3. These networks are simple and easily understandable.

3. Frame Representation

A frame is a record like structure which consists of a collection of attributes and its values to describe an entity in the world. Frames are the AI data structure which divides knowledge into substructures by representing stereotypes situations. It consists of a collection of slots and slot values. These slots may be of any type and sizes. Slots have names and values which are called facets.

Facets: The various aspects of a slot is known as **Facets**. Facets are features of frames which enable us to put constraints on the frames. Example: IF-NEEDED facts are called when data of any particular slot is needed. A frame may consist of any number of slots, and a slot may include any number of facets and facets may have any number of values. A frame is also known as **slot-filter knowledge representation** in artificial intelligence.

Frames are derived from semantic networks and later evolved into our modern-day classes and objects. A single frame is not much useful. Frames system consist of a collection of frames which are connected. In the frame, knowledge about an object or event can be stored together in the knowledge base. The frame is a type of technology which is widely used in various applications including Natural language processing and machine visions.

Example: 1

Let's take an example of a frame for a book

Slots	Filters
Title	Artificial Intelligence

Genre	Computer Science
Author	Peter Norvig
Edition	Third Edition
Year	1996
Page	1152

Example 2:

Let's suppose we are taking an entity, Peter. Peter is an engineer as a profession, and his age is 25, he lives in city London, and the country is England. So following is the frame representation for this:

Slots	Filter
Name	Peter
Profession	Doctor
Age	25
Marital status	Single
Weight	78

Advantages of frame representation:

1. The frame knowledge representation makes the programming easier by grouping the related data.
2. The frame representation is comparably flexible and used by many applications in AI.
3. It is very easy to add slots for new attribute and relations.
4. It is easy to include default data and to search for missing values.
5. Frame representation is easy to understand and visualize.

Disadvantages of frame representation:

1. In frame system inference mechanism is not be easily processed.
2. Inference mechanism cannot be smoothly proceeded by frame representation.
3. Frame representation has a much generalized approach.

4. Production Rules

Production rules system consist of (**condition, action**) pairs which mean, "If condition then action". It has mainly three parts:

- The set of production rules
- Working Memory
- The recognize-act-cycle

In production rules agent checks for the condition and if the condition exists then production rule fires and corresponding action is carried out. The condition part of the rule determines which rule may be applied to a problem. And the action part carries out the associated problem-solving steps. This complete process is called a recognize-act cycle.

The working memory contains the description of the current state of problems-solving and rule can write knowledge to the working memory. This knowledge match and may fire other rules.

If there is a new situation (state) generates, then multiple production rules will be fired together, this is called conflict set. In this situation, the agent needs to select a rule from these sets, and it is called a conflict resolution.

Example:

- **IF (at bus stop AND bus arrives) THEN action (get into the bus)**
- **IF (on the bus AND paid AND empty seat) THEN action (sit down).**
- **IF (on bus AND unpaid) THEN action (pay charges).**
- **IF (bus arrives at destination) THEN action (get down from the bus).**

Advantages of Production rule:

1. The production rules are expressed in natural language.
2. The production rules are highly modular, so we can easily remove, add or modify an individual rule.

Disadvantages of Production rule:

1. Production rule system does not exhibit any learning capabilities, as it does not store the result of the problem for the future uses.

2. During the execution of the program, many rules may be active hence rule-based production systems are inefficient.

Constraint Satisfaction Problems in Artificial Intelligence

We have encountered a wide variety of methods, including adversarial search and instant search, to address various issues. Every method for issue has a single purpose in mind: to locate a remedy that will enable that achievement of the objective. However there were no restrictions just on bots' capability to resolve issues as well as arrive at responses in adversarial search and local search, respectively.

These section examines the constraint optimization methodology, another form or real concern method. By its name, constraints fulfilment implies that such an issue must be solved while adhering to a set of restrictions or guidelines.

Whenever a problem is actually variables comply with stringent conditions of principles, it is said to have been addressed using the solving multi - objective method. Wow what a method results in a study sought to achieve of the intricacy and organization of both the issue.

Three factors affect restriction compliance, particularly regarding:

- It refers to a group of parameters, or X.
- D: The variables are contained within a collection several domain. Every variables has a distinct scope.
- C: It is a set of restrictions that the collection of parameters must abide by.

In constraint satisfaction, domains are the areas wherein parameters were located after the restrictions that are particular to the task. Those three components make up a constraint satisfaction technique in its entirety. The pair "scope, rel" makes up the number of something like the requirement. The scope is a tuple of variables that contribute to the restriction, as well as rel is indeed a relationship that contains a list of possible solutions for the parameters should assume in order to meet the restrictions of something like the issue.

Issues with Contains A certain amount Solved

For a constraint satisfaction problem (CSP), the following conditions must be met:

- States area
- fundamental idea while behind remedy.

The definition of a state in phase space involves giving values to any or all of the parameters, like as

X1 = v1, X2 = v2, etc.

There are 3 methods to economically beneficial to something like a parameter:

1. Consistent or Legal Assignment: A task is referred to as consistent or legal if it complies with all laws and regulations.
2. Complete Assignment: An assignment in which each variable has a number associated to it and that the CSP solution is continuous. One such task is referred to as a completed task.
3. A partial assignment is one that just gives some of the variables values. Projects of this nature are referred to as incomplete assignment.

Domain Categories within CSP

The parameters utilize one of the two types of domains listed below:

- Discrete Domain: This limitless area allows for the existence of a single state with numerous variables. For instance, every parameter may receive a endless number of beginning states.
- It is a finite domain with continous phases that really can describe just one area for just one particular variable. Another name for it is constant area.

Types of Constraints in CSP

Basically, there are three different categories of limitations in regard towards the parameters:

- Unary restrictions are the easiest kind of restrictions because they only limit the value of one variable.
- Binary resource limits: These restrictions connect two parameters. A value between x1 and x3 can be found in a variable named x2.
- Global Resource limits: This kind of restriction includes a unrestricted amount of variables.

The main kinds of restrictions are resolved using certain kinds of resolution methodologies:

- In linear programming, when every parameter carrying an integer value only occurs in linear equation, linear constraints are frequently utilised.
- Non-linear Constraints: With non-linear programming, when each variable (an integer value) exists in a non-linear form, several types of restrictions were utilised.

Note: The preferences restriction is a unique restriction that operates in the actual world.

Think of a Sudoku puzzle where some of the squares have initial fills of certain integers.

You must complete the empty squares with numbers between 1 and 9, making sure that no rows, columns, or blocks contains a recurring integer of any kind. This solving multi - objective issue is pretty elementary. A problem must be solved while taking certain limitations into consideration.

The integer range (1-9) that really can occupy the other spaces is referred to as a domain, while the empty spaces themselves were referred as variables. The values of the variables are drawn first from realm. Constraints are the rules that determine how a variable will select the scope.

Several issues exist when it comes to knowledge representation in artificial intelligence. These challenges can impact how effectively systems can capture, store, and manipulate knowledge. Some key issues include:

1. Expressiveness:

- Representational languages may struggle to express complex or nuanced relationships and knowledge, limiting the depth and accuracy of representation.

2. Scalability:

- As knowledge bases grow, scalability becomes a significant concern. Systems need to handle a vast amount of information efficiently.

3. Ambiguity and Uncertainty:

- Real-world knowledge often contains ambiguous or uncertain information, making it challenging to represent and reason with certainty.

4. Context Sensitivity:

- Knowledge can be highly context-dependent, requiring systems to understand and adapt to varying contexts.

5. Dynamic Environments:

- Representing knowledge in dynamic environments, where information changes frequently, poses a challenge. Systems need to adapt to new information effectively.

6. Inference and Reasoning:

- Performing effective inference and reasoning on represented knowledge can be complex, especially when dealing with incomplete or contradictory information.

7. Integration of Different Knowledge Types:

- Integrating diverse types of knowledge (textual, visual, sensor data, etc.) into a coherent representation can be difficult.

8. Common Sense Knowledge:

- Capturing and utilizing common-sense knowledge, which is crucial in human reasoning, remains a challenge in AI systems.

9. Computational Complexity:

- Some representation techniques may lead to computationally expensive operations, especially as the knowledge base grows.

10. Maintenance and Evolution:

- Managing, updating, and evolving knowledge bases over time requires efficient strategies to avoid knowledge obsolescence

will breadth fs always find optimal solution

Breadth-First Search (BFS) is a graph traversal algorithm that systematically explores all the neighbor nodes at the present depth before moving on to nodes at the next level of depth. In the context of finding solutions in a graph or tree, BFS guarantees that it will find the shallowest solution (the solution with the fewest edges or levels from the start node).

Optimal Solution Guarantee of BFS:

- For unweighted graphs and trees where all edge costs are equal, BFS will always find the shortest path from the starting node to any other reachable node. This holds true because BFS expands nodes level by level, ensuring that the first occurrence of a node in the search is the shortest path to that node.

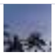
Limitations of BFS in Finding Optimal Solutions:

- **Weighted Graphs:** In the case of graphs where edges have different costs or weights, BFS may not find the optimal solution. BFS doesn't consider the weight of edges, so if there are varying edge costs, it might not find the shortest path.
- **Infinite Graphs:** In cases of infinite graphs or graphs with loops, BFS might not terminate or find a solution due to its exhaustive nature.

Summary:

BFS guarantees optimality in unweighted graphs where all edges have the same cost. However, in weighted graphs or scenarios with infinite graphs, its optimality cannot be guaranteed.

For scenarios with weighted graphs, other algorithms like Dijkstra's algorithm (for non-negative edge weights) or A* search (for pathfinding with heuristics) are better suited to find optimal solutions. These algorithms consider edge costs and incorporate heuristic information to find optimal paths efficiently.



You

define term heuristic list and describe various problems characterstics that needed to be considered for selecting appropriate heuristic for a given class of problems

ChatGPT

ChatGPT

A heuristic is a problem-solving strategy or technique that uses practical and intuitive methods to find solutions, especially when an exhaustive search or optimal solution is not feasible due to computational limitations. Heuristics aim to efficiently find near-optimal solutions or reduce search space without guaranteeing optimality.

Characteristics for Selecting Appropriate Heuristics:

1. **Admissibility:**
   - **Admissible Heuristic:** Estimates the lowest possible cost to reach the goal without overestimating. It never overestimates the true cost. Admissible heuristics guarantee optimality in algorithms like A* if they're admissible.
2. **Consistency:**
   - **Consistent Heuristic:** Also known as monotonic, it satisfies the triangle inequality in pathfinding problems. If h(n) is the estimated cost from node n to the goal and c(n, a, n') is the cost of moving from node n to n' using action a, then h(n) ≤ c(n, a, n') + h(n').
3. **Accuracy and Precision:**
   - **Accurate Heuristic:** Reflects the actual cost to reach the goal. More accurate heuristics tend to improve the quality of solutions.
4. **Problem-Specific Knowledge:**
   - Heuristics often leverage domain-specific knowledge. A good heuristic should incorporate information about the problem domain to guide the search effectively.
5. **Computational Efficiency:**
   - Heuristics should be computationally efficient to be practical. They should aid in reducing search space without significantly increasing computational complexity.
6. **Applicability:**

- The heuristic should be applicable to the specific problem or class of problems being addressed. Not all heuristics are suitable for every problem domain.

7. **Domain-Independence vs. Domain-Specificity:**
   - Some heuristics are general-purpose and can be applied across various domains. Others are tailored to specific problem domains and might not generalize well.
8. **Search Space Reduction:**
   - Effective heuristics narrow down the search space significantly, reducing the number of nodes or states to be explored.
9. **Scalability:**
   - Heuristics should ideally scale well with problem complexity and size, allowing efficient solutions for larger instances of the problem.
10. **Memory and Storage:**

- Heuristics that require excessive memory or storage might not be practical for real-world problems.

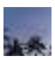
Selecting an appropriate heuristic often involves a trade-off between accuracy, computational efficiency, and problem-specific knowledge. The choice of heuristic greatly influences the efficiency and effectiveness of heuristic-based algorithms like A*, IDA*, or other problem-solving techniques in AI.

Heuristic problems are those for which finding an optimal solution might be computationally infeasible or highly complex. These problems often benefit from heuristic approaches that aim to find near-optimal solutions efficiently. Some heuristic problems include:

1. **Traveling Salesman Problem (TSP):**
   - Given a list of cities and the distances between them, find the shortest possible route that visits each city exactly once and returns to the starting city.
2. **Knapsack Problem:**
   - Given a set of items, each with a weight and a value, determine the items to include in a knapsack of a fixed capacity to maximize the total value.
3. **Scheduling Problems:**
   - Job scheduling, task scheduling, and resource allocation problems where tasks or jobs need to be assigned to resources subject to constraints while optimizing criteria like completion time or resource utilization.
4. **Graph Coloring Problem:**
   - Assign colors to vertices of a graph such that no two adjacent vertices have the same color, using the fewest possible colors.
5. **Planning and Route Optimization:**
   - Optimal pathfinding in maps or networks, considering factors like distance, time, or cost while navigating through multiple points or constraints.
6. **Machine Learning Model Optimization:**
   - Tuning hyperparameters of machine learning models to maximize performance (accuracy, precision, recall, etc.) without exhaustive search.

7. **Packing Problems:**
    - Container loading, bin packing, or cutting stock problems where items of various shapes and sizes need to be packed into containers efficiently.
8. **Vehicle Routing Problem (VRP):**
    - Efficiently plan routes for vehicles to deliver goods to multiple locations while minimizing total travel distance or time.
9. **Facility Location Problems:**
    - Determining optimal locations for facilities like warehouses, hospitals, or distribution centers to serve a given set of demand points.
10. **Constraint Satisfaction Problems (CSPs):**
- Problems where variables need to be assigned values subject to constraints, such as Sudoku, N-Queens, or scheduling problems.

Heuristic methods, such as greedy algorithms, local search, or metaheuristic approaches like genetic algorithms, simulated annealing, and ant colony optimization, are often employed to tackle these problems efficiently by finding good approximate solutions.



You

components to define an ai problem

ChatGPT

ChatGPT

Defining an AI problem involves breaking it down into several key components that help formalize the nature of the problem and guide the development of a suitable solution. Here are the essential components:

1. **Initial State:**
    - It represents the starting configuration or state of the problem. It defines where the problem-solving process begins.
2. **Goal State or Objective:**
    - Specifies the desired outcome or the goal that the AI system aims to achieve. It defines the conditions under which the problem is considered solved.
3. **Actions/Operators:**
    - Represents the set of possible moves or actions available to the AI agent. These actions transition the agent from one state to another toward the goal.
4. **Transition Model/Successor Function:**
    - Describes the result of applying an action in a particular state. It defines how the state changes when an action is performed.
5. **State Space:**
    - Encompasses all possible states that can be reached from the initial state by taking a sequence of actions.
6. **Cost Function:**
    - Assigns a cost or value to each action or state transition. It quantifies the "cost" or "distance" between states and helps the AI agent to choose more optimal paths or solutions.
7. **Constraints:**
    - Defines any restrictions or limitations that must be adhered to while solving the problem. Constraints can be related to time, resources, rules, or specific conditions.
8. **Environment/Dynamic Aspects:**

- Describes the environment in which the AI agent operates. It includes any dynamic or changing aspects that affect the state transitions or the problem-solving process.

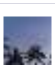
9. **Uncertainty or Stochasticity (if applicable):**
   - Considers any randomness or uncertainty in the problem, which may require the AI system to make decisions under uncertainty.
10. **Performance Metrics:**
    - Specifies how the performance or quality of the solution is measured. It provides a criterion to evaluate the effectiveness of the AI solution.

Search algorithms can be effectively utilized to solve constraint satisfaction problems (CSPs) like the N-Queens problem, which involves finding a way to place N chess queens on an N×N chessboard so that no two queens threaten each other. Here's how search can be applied:

1. **Problem Representation:**
   - Represent the problem state: In the N-Queens problem, each state is a placement configuration of queens on the board, where each queen occupies a unique row and column.
2. **Constraints Representation:**
   - Encode constraints: Define constraints that limit the queen placements, such as no two queens being able to attack each other horizontally, vertically, or diagonally.
3. **Search Algorithm:**
   - Utilize search algorithms: Algorithms like backtracking, depth-first search (DFS), or constraint propagation techniques (e.g., forward checking, arc consistency) can be employed to systematically explore the search space of possible queen placements while adhering to the constraints.
4. **State Space Representation:**
   - Generate successor states: At each step of the search, generate successor states by placing a queen in a valid position that satisfies the constraints. This process involves exploring possible queen placements in different columns, recursively moving through the search space.
5. **Pruning and Backtracking:**
   - Implement pruning strategies: Apply techniques like constraint propagation to reduce the search space by eliminating configurations that violate constraints. Backtracking is employed when a partial placement leads to a dead-end, allowing the algorithm to backtrack to the last valid state and explore alternative paths.
6. **Goal Test:**
   - Define the goal state: Reach a state where all N queens are placed on the board without threatening each other, satisfying all constraints. Once a valid configuration is found, the problem is solved.

For the N-Queens problem, a typical approach involves a backtracking-based search algorithm, where the placement of queens is attempted incrementally, and if a placement violates any constraint, the algorithm backtracks to explore other possible configurations.

By systematically exploring the search space while respecting the constraints, search algorithms can efficiently find solutions to constraint satisfaction problems like the N-Queens problem, ensuring that all constraints are met in the final solution configuration.



You

what difficulties arise when these problems become extremly large and methods to solve large problems efficently

ChatGPT

ChatGPT

When constraint satisfaction problems (CSPs) become extremely large, several difficulties arise:

1. **Exponential Increase in Search Space:**
   - The number of possible configurations or states grows exponentially with the problem size. This explosion in the search space makes exhaustive search methods impractical or infeasible.
2. **Computational Complexity:**
   - The computational effort required to evaluate each state or configuration becomes overwhelming as the problem size increases. Traditional algorithms might take an inordinate amount of time to find solutions.
3. **Memory and Processing Constraints:**
   - Storing and processing a large number of states might exceed memory or processing capabilities, leading to performance bottlenecks or system limitations.
4. **Constraint Propagation Inefficiency:**
   - Constraint propagation techniques might become less effective as the problem grows larger, leading to a decreased ability to eliminate potential solutions efficiently.

To tackle these challenges efficiently, several methods and strategies can be employed:

1. **Heuristic Search Algorithms:**
   - Implement heuristic-based search methods like local search algorithms (e.g., simulated annealing, genetic algorithms) that prioritize exploration of more promising regions in the search space, potentially avoiding exhaustive exploration.
2. **Constraint Relaxation and Approximation:**
   - Relaxing some constraints or using approximations to reduce the problem complexity can help in finding solutions faster. Approximate methods sacrifice optimality for computational efficiency.
3. **Parallel and Distributed Computing:**
   - Utilize parallel computing or distributed systems to divide the problem into smaller subproblems and solve them concurrently, speeding up the search process.
4. **Intelligent Pruning Techniques:**
   - Develop more sophisticated pruning techniques or constraint propagation methods specifically tailored for large-scale problems to reduce the search space effectively.
5. **Hybrid Approaches:**
   - Combine different algorithms or approaches (hybridization) to leverage the strengths of multiple methods. For instance, combining constraint propagation with heuristic search methods.
6. **Domain-Specific Knowledge:**
   - Incorporate domain-specific knowledge or problem-specific heuristics to guide the search process more intelligently, focusing on promising regions of the solution space.
7. **Incremental Problem Solving:**
   - Solve the problem incrementally, breaking it down into smaller, manageable parts or focusing on specific subsets of the problem space to find partial solutions.

By employing these strategies and adapting algorithms to handle the complexities of large-scale CSPs, it becomes possible to navigate the extensive solution space more efficiently, allowing for the exploration of potential solutions within a reasonable timeframe and computational resources.

When choosing a search algorithm for a problem, several characteristics need consideration to ensure an efficient and effective solution strategy:

1. **Completeness:**

- Does the algorithm guarantee finding a solution if one exists? Complete algorithms ensure that a solution will be found given enough time and resources.

2. **Optimality:**
   - Does the algorithm always find the optimal solution? Optimal algorithms guarantee finding the best (most desirable) solution among all possible solutions.
3. **Time Complexity:**
   - How long does the algorithm take to find a solution? Assess the algorithm's efficiency in terms of time required for different problem sizes.
4. **Space Complexity:**
   - How much memory or space does the algorithm use? Consider the memory requirements of the algorithm, especially for larger problem instances.
5. **Scalability:**
   - Does the algorithm's performance degrade significantly as the problem size increases? An algorithm should remain efficient even as the problem scales.
6. **Heuristics:**
   - Can the algorithm benefit from domain-specific heuristics or knowledge? Some algorithms can exploit heuristics to guide the search more intelligently.
7. **Admissibility of Heuristics (for informed search):**
   - In algorithms that use heuristics (like A*), are these heuristics admissible? Admissible heuristics never overestimate the true cost to reach the goal.
8. **Memory Requirements:**
   - How much memory does the algorithm need to store data structures or search states? Assess the trade-off between memory consumption and search efficiency.
9. **Determinism:**
   - Does the algorithm produce the same results for the same problem instance each time it's run? Deterministic algorithms produce consistent outcomes.
10. **Parallelizability:**
    - Can the algorithm be easily parallelized or distributed across multiple processors or systems for faster computation?
11. **Constraint Handling:**
    - Is the algorithm capable of handling constraints efficiently, especially in constraint satisfaction problems (CSPs)?
12. **Robustness:**
    - How well does the algorithm perform under various conditions, including noisy or uncertain environments?
13. **Implementation Complexity:**
    - How complex is the algorithm to implement? Some algorithms may be more straightforward to code or adapt to specific problems than others.

---

various types of non monotonic logics

ChatGPT

ChatGPT

Non-monotonic logics are formal systems that depart from classical logic by allowing conclusions to be retracted or modified in the presence of new information. Here are some types of non-monotonic logics:

1. **Default Logic:**
   - Default logic introduces default rules that capture typical assumptions or defaults. These rules are used to derive conclusions unless overridden by explicit information.
2. **Autoepistemic Logic:**
   - Autoepistemic logic deals with self-referential statements about knowledge and beliefs. It allows for reasoning about what an agent knows and believes about its own knowledge.
3. **Circumscription:**
   - Circumscription aims to minimize exceptions by postulating minimal worlds where exceptions to general rules occur. It tries to provide explanations without excessive changes to existing knowledge.
4. **Relevance Logic:**
   - Relevance logic focuses on logical systems where the inference relation is sensitive to the structure of premises and conclusions. It aims to maintain relevance between premises and conclusions.
5. **Paraconsistent Logic:**
   - Paraconsistent logic deals with inconsistencies in reasoning by allowing for the existence of contradictions without trivializing the logic or leading to every statement being provable.
6. **Modal Non-Monotonic Logics:**
   - Modal logics extend classical logic by introducing modal operators (e.g., necessity, possibility) and are used to reason about necessity and possibility in a non-monotonic context.
7. **Fuzzy Logic:**
   - Fuzzy logic extends classical binary logic by allowing degrees of truth between 0 and 1, enabling reasoning in situations where boundaries between true and false are not clear-cut.

Linguistic variables are a concept in fuzzy logic used to represent qualitative terms or labels in a quantitative and mathematically precise manner. They allow for the expression of imprecise or vague concepts in a way that a computer or an automated system can interpret and process.

In traditional mathematics and logic, variables usually take specific numerical values. However, in real-world scenarios, many concepts are not easily quantifiable. Linguistic variables provide a way to describe these concepts using linguistic terms or labels, such as "low," "medium," "high," "very hot," "cold," etc.

Linguistic variables consist of three main components:

1. **Term Set:**
   - A linguistic variable consists of a set of terms or labels that describe its values. For example, a linguistic variable "temperature" might have terms like "cold," "warm," and "hot."
2. **Membership Functions:**
   - Each term in a linguistic variable is associated with a membership function that assigns a degree of membership to elements of the universe of discourse (the range of possible values). This degree represents the degree to which an element belongs to a particular term. For instance, in the "temperature" variable, the term "warm" might have a membership function that assigns higher membership values to temperatures between 20°C and 30°C.
3. **Fuzzy Sets:**

- Linguistic variables use fuzzy sets to represent their values. A fuzzy set allows elements to have partial membership in different terms simultaneously. This means an element can be "slightly warm" and "slightly cold" at the same time.

In essence, linguistic variables enable the use of natural language terms to describe input and output variables in fuzzy logic systems. This facilitates human-like reasoning and decision-making in systems dealing with imprecise or uncertain information, allowing for more flexible and intuitive control in various applications such as fuzzy controllers, expert systems, and linguistic modeling.

---

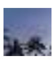
1. **Union (Logical OR):**
   - The union of two fuzzy sets A and B results in a fuzzy set that represents the degree to which elements belong to either set. It's denoted by $A \cup B$ and is calculated by taking the maximum membership value for each element from both sets.
2. **Intersection (Logical AND):**
   - The intersection of two fuzzy sets A and B results in a fuzzy set that represents the degree to which elements belong to both sets. It's denoted by $A \cap B$ and is calculated by taking the minimum membership value for each element from both sets.
3. **Complement (Negation):**
   - The complement of a fuzzy set A represents the degree to which elements do not belong to set A. It's denoted by $\neg A$ or A' and is calculated by subtracting the membership values from 1.
4. **Difference:**
   - The difference between fuzzy sets A and B results in a fuzzy set that represents the degree to which elements belong to set A but not to set B. It's denoted by $A - B$ and is calculated as the minimum of $A(x)$ and $1 - B(x)$, where x is an

Hierarchical and conditional planning are two different approaches in artificial intelligence used to structure planning problems. Let's compare them with examples:

1. **Hierarchical Planning:**
   - **Definition:** Hierarchical planning involves breaking down a complex problem into smaller, more manageable sub-problems organized in a hierarchical structure.
   - **Example:** Consider the task of organizing an event, like a conference. In hierarchical planning:
     - **Top-Level Plan:** The highest level involves major tasks like scheduling, venue selection, and inviting speakers.

        - **Mid-Level Plans:** Each major task breaks down into sub-tasks. For instance, scheduling involves sub-tasks like arranging time slots, allocating rooms, and coordinating with speakers.
        - **Low-Level Plans:** These sub-tasks further decompose into detailed actions, such as contacting individual speakers, arranging equipment, and printing materials.
2. **Conditional Planning:**
    - **Definition:** Conditional planning deals with uncertainty by reasoning about action sequences to achieve goals in various possible situations or conditions.
    - **Example:** Let's say a robot has to navigate a room to find an object. In conditional planning:
        - **If-Then Conditions:** The robot considers various conditions it might encounter, like finding the object on a table, under a chair, or behind a door.
        - **Plans for Each Condition:** For each condition, the robot devises a plan. If the object is on a table, it plans to reach the table and grab the object. If it's under a chair, it plans to navigate to the chair and retrieve the object.
        - **Adaptive Action Selection:** The robot adapts its actions based on the perceived condition (e.g., if the object isn't on the table, it moves to search elsewhere).

In summary, hierarchical planning organizes tasks into a structured hierarchy, while conditional planning deals with uncertainties by creating plans for different possible conditions or situations. The hierarchical approach focuses on breaking down complex tasks, while conditional planning deals with decision-making in the face of uncertainty or varying conditions. Both approaches serve different purposes and are applied based on the nature of the problem and the level of uncertainty involved.



You

role of intelligent agents

ChatGPT

ChatGPT

Intelligent agents play a pivotal role in artificial intelligence and computer science. These agents are software entities that perceive their environment, reason about the information received, and take actions to achieve specific goals or tasks. Their role spans various domains and applications:

1. **Automation and Autonomy:**
    - Intelligent agents automate tasks, processes, and decision-making, reducing human intervention.
    - They operate autonomously, making decisions based on predefined rules or learning from data.
2. **Problem Solving and Decision Making:**
    - Agents solve complex problems by analyzing available data and selecting the most suitable actions to achieve desired outcomes.
    - They employ reasoning mechanisms to make informed decisions in uncertain or dynamic environments.
3. **Adaptability and Learning:**
    - Intelligent agents adapt to changing environments or circumstances by modifying their behavior or strategies.
    - They learn from experience, improving their performance over time through machine learning or other learning techniques.
4. **Interaction and Communication:**
    - Agents interact with users or other systems, responding to queries, providing information, and performing tasks.
    - They communicate with other agents or entities, exchanging data and coordinating actions.
5. **Sensing and Perception:**
    - Agents sense and perceive their environment through sensors or data inputs, interpreting information to understand their surroundings.

- They use this perception to make decisions and take appropriate actions.

6. **Efficiency and Optimization:**
   - Intelligent agents optimize processes, resources, and workflows, aiming for efficiency and resource utilization.
   - They analyze data to identify patterns or trends, contributing to better decision-making and problem-solving.
7. **Support in Various Domains:**
   - In various fields like finance, healthcare, manufacturing, and transportation, intelligent agents assist in decision support, predictive analysis, and automation of tasks.
8. **Continuous Improvement:**
   - Agents often incorporate feedback loops, allowing them to learn from mistakes or successes, leading to continuous improvement and adaptation.

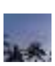
spect	Hierarchical Planning	Conditional Planning
Definition	Hierarchical planning involves breaking down a complex problem into a hierarchy of smaller, more manageable subproblems.	Conditional planning is based on making decisions sequentially, where each decision depends on certain conditions or states.
Organization	Organizes planning into multiple levels or layers, often representing a top-down approach with higher-level goals guiding lower-level actions.	Sequential decision-making is followed, where actions are taken based on conditions or states encountered in a defined sequence.
Decomposition	Divides the problem into subproblems, each of which may have its own planning procedure, goals, and constraints.	Plans actions based on specific conditions, often involving if-then rules or state-action pairs.
Complexity Handling	Manages complexity by breaking it into manageable modules or layers, each dealing with a specific aspect of the problem.	Handles complexity by making decisions at each step based on conditions, reducing the problem into a sequence of conditional actions.
Flexibility	Offers flexibility in addressing different levels of abstraction and granularity in planning, allowing for different levels of detail.	Provides flexibility in making decisions based on changing conditions or states encountered during execution.
Coordination	Requires coordination and integration among various levels or layers of the hierarchy to ensure the achievement of higher-level goals.	Coordinates actions based on conditions or states encountered during execution, ensuring the fulfillment of specific conditions or reaching certain states.
Examples	In robotics, hierarchical planning may involve high-level goals (navigate to a location) broken down into subgoals (avoid obstacles, follow a path).	Conditional planning in game AI might involve decision trees where actions are chosen based on game states (if player health is low, retreat; if enemy is near, attack).

spect	Hierarchical Planning	Conditional Planning
Applicability	Suited for problems with multiple levels of abstraction or complexity, allowing modularity and easy management of various layers.	Appropriate for problems where actions depend on specific conditions or states encountered during execution, enabling decision-making based on changing circumstances.

Certainly! PEAS stands for Performance measure, Environment, Actuators, and Sensors. Let's apply this framework to medical diagnosis systems:

1. **Performance Measure:**
    - The performance measure for a medical diagnosis system would be the accuracy and efficiency in correctly identifying a patient's illness or condition. It involves how well the system diagnoses diseases, recommends treatments, and predicts outcomes.
2. **Environment:**
    - The environment for a medical diagnosis system includes the patient, medical records, symptoms, test results, and possibly external factors such as environmental conditions or epidemiological data. It's the context in which the system operates and gathers information.
3. **Actuators:**
    - Actuators in a medical diagnosis system could include various components such as:
        - **Treatment Recommendation Systems:** Suggesting appropriate medications or therapies based on the diagnosis.
        - **Alert Systems:** Notifying healthcare providers of critical diagnoses or urgent conditions.
        - **Data Logging Systems:** Recording patient information, test results, and treatment plans in electronic health records.
4. **Sensors:**
    - Sensors in this context refer to the means by which the system gathers information or inputs. They might include:
        - **Patient Symptoms:** Collected through patient interviews or interactions.
        - **Medical Tests:** Data obtained from blood tests, imaging scans, genetic tests, etc.
        - **Historical Data:** Past medical records, family medical history, and other relevant patient information.

Aspect	Non-Monotonic Expert Systems	Decision Tree-based Expert Systems
Working	- Utilizes default logic or defeasible reasoning.	- Constructs a tree-like model of decisions and consequences.
Reasoning Approach	- Accommodates changing conclusions based on new information.	- Top-down recursive reasoning based on input features.
Handling Uncertainty	- Adapts to uncertainty and inconsistency in the knowledge base.	- Generally assumes a more deterministic approach.
Revision of Conclusions	- Conclusions can be revised in the presence of conflicting data.	- Follows fixed decision paths, less amenable to revision.
Characteristics	- Flexibility in handling dynamic and evolving knowledge.	- Transparent decision-making process with easily interpretable rules.
Knowledge Representation	- Dynamic knowledge representation allows for updates.	- Static representation based on the decision tree structure.
Applications	- Suited for dynamic domains like medical diagnosis or fault detection.	- Commonly used in finance, healthcare, and classification tasks.
Handling Inconsistencies	- Designed to handle conflicting or incomplete information.	- May struggle with inconsistencies, as decisions are rule-based.
Interpretability	- May be less transparent due to the dynamic nature of conclusions.	- Transparent and easily interpretable due to the tree structure.
Decision Revision Mechanism	- Utilizes default logic or defeasible reasoning to revise decisions.	- Typically follows a fixed set of rules without dynamic revision.



You

planning with propositional logic in detail

ChatGPT

ChatGPT

Planning with propositional logic involves representing the states of a problem using logical propositions and applying inference to determine a sequence of actions that leads from an initial state to a desired goal state.

Components Involved:

1. **State Representation:**
    - **State Variables:** Represent the components of the problem domain using propositions. For example, in a blocks world, propositions might represent the positions of blocks, like On(A, B) (Block A is on Block B) or Clear(A) (Block A is clear).
2. **Actions:**
    - **Preconditions:** Represented as logical formulas specifying the conditions that must be true for an action to be applicable. For instance, for the action Move(A, X, Y) (Move block A from X to Y), preconditions might include Clear(A), On(A, X), and Clear(Y).
    - **Effects:** Describe the changes in state resulting from the execution of an action. For the Move action, effects might include ¬On(A, X), ¬Clear(Y), and On(A, Y).

Planning Process:

1. **Initial State:** Represented using a set of propositions that describe the starting configuration of the problem domain.
2. **Goal State:** Defined by a set of propositions representing the desired final configuration or state of the problem.
3. **Search Algorithms:** Employed to find a sequence of actions transforming the initial state into the goal state.
    - **Forward Chaining:** Starts with the initial state and applies actions until the goal state is achieved. It systematically explores actions based on their applicability.
    - **Backward Chaining:** Begins with the goal state and works backward, attempting to find actions that can achieve the current state from a previous state. It reasons backward from the goal to the initial state.
4. **Inference:** Utilized to determine which actions can be applied in the current state, based on the applicable preconditions of available actions.

Example:

Consider a blocks world problem where the goal is to stack Block A onto Block B.

- **Initial State:** On(A, Table), On(B, Table), Clear(A), Clear(B), ArmEmpty
- **Goal State:** On(A, B)

Actions available:

1. Move(A, Table, B)
2. Move(A, Table, C)
3. Move(A, B, Table)
4. Move(A, C, Table)
5. Move(A, B, C)
6. Move(A, C, B)

Using propositional logic, a planning algorithm could explore these actions by checking their preconditions against the current state and applying those actions whose preconditions hold true, aiming to reach the desired goal state.

Planning with propositional logic provides a structured framework for reasoning about action and state transitions in a problem domain, making it a fundamental approach in AI planning systems. However, it's limited in its ability to handle complex domains with large state spaces or continuous variables.

Certainly! Rule-based systems and decision tree-based expert systems are both widely used in AI for knowledge representation and problem-solving. Here's a comparison between the two:

Rule-Based Systems:

1. **Representation:**
    - **Knowledge Representation:** Utilizes a set of if-then rules to encode knowledge.
    - **Rule Structure:** Comprises rules in the form of conditional statements (if condition, then action) or production rules.
2. **Inference Mechanism:**
    - **Forward Chaining:** Applies rules to data or facts until a conclusion or action is reached.
    - **Backward Chaining:** Begins with a goal and works backward to determine what facts or data are needed to satisfy that goal.
3. **Interpretability:**
    - **Human Readability:** Rule-based systems are generally human-readable and comprehensible as they follow a logical format.
4. **Complexity Handling:**
    - **Handling Complexity:** Can become complex when managing numerous rules or handling conflicting rules.
5. **Domain Suitability:**
    - **Suitable Domains:** Suitable for domains where knowledge can be explicitly defined and represented using rules.

Decision Tree-Based Systems:

1. **Representation:**
    - **Knowledge Representation:** Uses a tree-like structure to represent decisions and their potential consequences.
    - **Tree Nodes:** Nodes represent decision points based on attributes, and branches signify possible outcomes.
2. **Inference Mechanism:**
    - **Tree Traversal:** Evaluates attributes from the root to the leaf nodes, making decisions based on attribute values.
3. **Interpretability:**
    - **Human Readability:** Decision trees are relatively easy to interpret and visualize, offering a clear decision-making path.
4. **Complexity Handling:**
    - **Handling Complexity:** Prone to overfitting in complex datasets, especially when the tree becomes deep or includes too many attributes.
5. **Domain Suitability:**
    - **Suitable Domains:** Suitable for problems where attributes and their relationships are well-defined and can be represented hierarchically.

Comparison Summary:

- **Representation:** Rule-based systems use if-then rules, while decision trees use a branching structure based on attribute values.
- **Inference Mechanism:** Rule-based systems employ chaining mechanisms (forward/backward), while decision trees use attribute traversal for decision-making.
- **Interpretability:** Both systems are relatively interpretable, with decision trees providing a visual representation.
- **Complexity Handling:** Both can struggle with complexity, with decision trees prone to overfitting and rule-based systems to rule conflicts.

demerits of expert sysyem

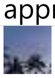
ChatGPT

ChatGPT

Expert systems, while powerful in many ways, also come with certain limitations and disadvantages:

1. **Knowledge Acquisition:** Acquiring and encoding knowledge into the system can be time-consuming and expensive, requiring expert input and ongoing updates.
2. **Limited Domain:** Expert systems are confined to the specific domain for which they are designed. They lack generalization beyond the defined knowledge base.
3. **Brittleness:** These systems can be fragile in handling unexpected or new scenarios outside their programmed knowledge, leading to errors or incorrect conclusions.
4. **Maintenance Challenges:** Expert systems require continuous maintenance and updates to stay relevant and accurate as knowledge evolves.
5. **Scalability:** Scaling up an expert system to cover broader or more complex domains can be challenging, often requiring restructuring or redesign.
6. **Lack of Common Sense:** They lack common sense reasoning and might struggle in contexts that require human intuition or tacit knowledge.
7. **Costly Development:** The initial development and implementation of expert systems can be costly due to the need for specialized expertise and resources.
8. **Interpretability and Explainability:** While they can make decisions, explaining the reasoning behind those decisions can be challenging, leading to issues with transparency.
9. **Dependency on Experts:** Expert systems heavily rely on experts for knowledge input and validation, making them sensitive to expert biases or limitations.
10. **Difficulty Handling Uncertainty:** Dealing with uncertain or incomplete information is a challenge for expert systems, as they often require clear, precise data to make decisions.

Despite these limitations, expert systems continue to be valuable in various fields, especially when well-designed and used within their intended scope and limitations. Advances in AI and machine learning have also led to hybrid systems that mitigate some of these drawbacks by combining the strengths of expert systems with data-driven approaches.



You

explain working of MYCIN expert system

ChatGPT

ChatGPT

MYCIN was an early expert system designed to assist in diagnosing bacterial infections and recommending antibiotics. Developed in the 1970s at Stanford University, MYCIN used a rule-based approach to mimic the decision-making process of infectious disease specialists. Here's a simplified overview of how MYCIN worked:

1. Knowledge Base:

MYCIN contained a vast knowledge base that included:

- **Facts:** Information about symptoms, patient history, laboratory test results, etc.
- **Rules:** Rules were structured as "IF-THEN" statements based on the expertise of infectious disease specialists.
- **Inference Engine:** A set of rules to infer conclusions from the facts provided.

2. Consultation Process:

1. **Data Entry:** The system would prompt the user (usually a physician) to input information about the patient's symptoms, medical history, and test results.
2. **Rule Application:** MYCIN's inference engine would then apply a set of rules based on the entered data to reach conclusions. For instance:
    - IF the patient has fever AND IF the blood culture is positive, THEN consider a bacterial infection.
    - IF there's a bacterial infection AND IF the bacteria are sensitive to a certain antibiotic, THEN recommend that antibiotic.
3. **Uncertainty Handling:** MYCIN also handled uncertainty by incorporating probabilistic reasoning. It used certainty factors (values between 0 and 1) to represent the system's confidence in specific conclusions.
4. **Explanation and Justification:** MYCIN was capable of explaining its reasoning process by providing justifications for its recommendations. It could explain why certain antibiotics were suggested based on the symptoms and test results.
5. **Recommendations:** Finally, MYCIN provided recommendations for antibiotic treatments along with their certainty factors.

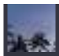
Advantages:

- MYCIN was groundbreaking for its time, showcasing the potential of expert systems in complex decision-making domains.
- It demonstrated the ability to emulate the expertise of human specialists in diagnosing and recommending treatments.

Limitations:

- **Domain-Specific:** MYCIN was designed specifically for diagnosing bacterial infections and was not adaptable to other medical conditions.
- **Expertise Limitation:** Its knowledge was confined to what was programmed, lacking the ability to learn or adapt beyond its predefined rules.
- **Handling Complexity:** While innovative, MYCIN struggled with the complexity of medical decision-making, especially in scenarios with overlapping symptoms or atypical cases.

Despite its limitations, MYCIN laid the foundation for subsequent expert systems and remains a significant milestone in the history of AI and medical informatics.



You

what is resolution and unification in ai

ChatGPT

ChatGPT

In artificial intelligence and logic programming, resolution and unification are fundamental concepts used in automated theorem proving and logical reasoning.

Unification:

- **Unification** is a process used to find substitutions for variables in logical expressions to make them equivalent.
- It's a mechanism to make two expressions match by assigning values to variables in such a way that both expressions become identical or unified.
- For example, given two expressions: P(x) and P(A), unification would involve finding a substitution (x/A) that would make these expressions equivalent.

Resolution:

- **Resolution** is a rule of inference used for proving the validity of logical statements or resolving contradictions in a logical system.
- It's a fundamental step in resolution-based theorem proving techniques, like in propositional and first-order logic.
- The resolution principle states that if you have two clauses (disjunctions of literals) containing complementary literals (a literal and its negation), you can resolve them to infer a new clause by removing the complementary literals.
- For instance, if you have clauses (P ∨ Q) and (¬Q), resolution allows you to infer a new clause (P).

Relationship between Unification and Resolution:

- Unification is often used as a subroutine in the resolution process. Before applying the resolution rule, the expressions involved are unified to determine if they can be made equivalent.
- Unification helps identify substitutions that make the resolution step possible. It finds a substitution that allows the resolution of clauses containing complementary liter

You have three jugs, measuring 12 gallons, 8 gallons, and 3 gallons, and a water faucet. You can fill the jugs up, empty them out from one to another or onto the ground. You need to measure out exactly one gallon. give complete space state and set of all applicable rules

ChatGPT

ChatGPT

The complete space state for this problem involves the possible configurations of water in the three jugs. Each state represents the amount of water in the 12-gallon, 8-gallon, and 3-gallon jugs.

Let's represent the jugs as A (12 gallons), B (8 gallons), and C (3 gallons). The initial state is (0, 0, 0) where all jugs are empty.

Complete Space State:

1. (0, 0, 0) - Initial state, all jugs empty.
2. (12, 0, 0) - 12-gallon jug filled.
3. (0, 8, 0) - 8-gallon jug filled.
4. (0, 0, 3) - 3-gallon jug filled.
5. (12, 8, 0) - 12-gallon and 8-gallon jugs filled.
6. (12, 0, 3) - 12-gallon and 3-gallon jugs filled.
7. (0, 8, 3) - 8-gallon and 3-gallon jugs filled.
8. (4, 8, 0) - 4 gallons in the 12-gallon jug (resulting from transferring from the 12-gallon jug to the 8-gallon jug).

Applicable Rules:

1. Fill a jug:
   - Fill the 12-gallon jug (A).
   - Fill the 8-gallon jug (B).
   - Fill the 3-gallon jug (C).
2. Empty a jug:
   - Empty the 12-gallon jug (A).
   - Empty the 8-gallon jug (B).
   - Empty the 3-gallon jug (C).
3. Pour water from one jug to another:
   - Pour from A to B.
   - Pour from B to A.
   - Pour from A to C.
   - Pour from C to A.
   - Pour from B to C.
   - Pour from C to B.

Note:

- These rules can be applied based on the current state and will generate new states until reaching the desired state of having exactly 1 gallon in one of the jugs. The rules help in manipulating the water in the jugs to achieve the target quantity of 1 gallon.

> Well for me the simplest way to do it would be;
>
> 1. Fill up the 8 gallon bucket and pour it into the 12 gallon bucket. You now have 8 gallons in the 12 gallon bucket.
> 2. Fill up the 8 gallon bucket another time and, again, pour it into the 12 gallon bucket until it . You now have 12 gallons in the 12 gallon bucket and 4 gallons left in the 8 gallon bucket.
> 3. Next, take the 8 gallon bucket, that now has 4 gallons in it, and pour it into the 3 gallon bucket until it is full.
> 4. By doing this, you will have the the requisite 1 gallon left in the 8 gallon bucket.

Hence, to get exactly 1 gallon of water:

1. fill completely the 12 gallons jug
2. pour the water from the 12-gallon jug into the 8-gallon jug until it is completely filled.
3. Pour the remaining water in the 12-gallon jug into the 3-gallon jug until it is completely filled.
4. The remaining water in the 12-gallon jug is exactly 1 gallon.

Certainly! Mean-End Analysis (MEA) is a problem-solving approach that breaks down the main goal into smaller subgoals and identifies the actions needed to bridge the gap between the current state and the desired goal state.

In the monkey-and-bananas problem:

Current State:

- The monkey is not near the box.
- The bananas are out of reach.
- The box is not positioned under the bananas.

Goal State:

- The monkey grasps the bananas.

MEA Steps:

1. **Identify the Differences:**
   - The monkey needs to get near the box.
   - The box needs to be moved under the bananas.
   - The monkey needs to climb on top of the box to reach the bananas.

2. **Define Subgoals:**
    - Subgoal 1: Get near the box.
    - Subgoal 2: Move the box under the bananas.
    - Subgoal 3: Climb on top of the box.
3. **Choose Actions to Achieve Subgoals:**
    - For Subgoal 1: The monkey needs to move towards the box.
    - For Subgoal 2: The monkey needs to push the box under the bananas.
    - For Subgoal 3: Once the box is under the bananas, the monkey climbs on top of it.
4. **Identify Obstacles:**
    - Obstacle for Subgoal 1: Monkey is not near the box.
    - Obstacle for Subgoal 2: Box is not positioned under the bananas.
    - Obstacle for Subgoal 3: The box is not accessible for climbing.
5. **Execute Actions:**
    - To get near the box, the monkey moves towards it.
    - Once near the box, the monkey pushes the box under the bananas.
    - With the box in position, the monkey climbs on top of it.
6. **Achieve the Goal State:**
    - Finally, the monkey, now on top of the box, is able to grasp the bananas, achieving the ultimate goal.

• John likes all kinds of food.

• Apples are food.

• Chicken is food.

• Anything anyone eats and isn't killed by is food.

• Bill eats peanuts and is still alive.

• Sue eats everything Bill eats.

i. Translate these sentences into formulas in FOPC.
ii. Convert the formulas into clause form.
iii. Use resolution to prove that John likes peanuts.
iv. Use resolution to answer the question, "What food does Sue eat?"
v.

(a) Translate these sentences into formulas in FOPC.

1. $\forall x\ Food(x) \rightarrow Likes(John, x)$

2. $Food(Apples)$

3. $Food(Chicken)$

4. $\forall x \exists y\ Eats(y, x) \wedge \neg KilledBy(y, x) \rightarrow Food(x)$

5. $Eats(Bill, Peanuts) \wedge \neg KilledBy(Bill, Peanuts)$

6. $\forall x\ Eats(Bill, x) \rightarrow Eats(Sue, x)$

(c) Convert the formulas of part (a) into clause form.

1. ¬F ood(x) ∨ Likes(John, x)
2. F ood(Apples)
3. F ood(Chicken)
4. ¬Eats(y, x) ∨ KilledBy(y, x) ∨ F ood(x)
5. Eats(Bill, P eanuts)
6. ¬KilledBy(Bill, P eanuts)
7. ¬Eats(Bill, x) ∨ Eats(Sue, x)

(d) Use resolution to prove that John likes peanuts.

8. ¬Likes(John, P eanuts) negated conclusion
9. ¬F ood(P eanuts) 1 and 8
10. ¬Eats(y, P eanuts) ∨ KilledBy(y, P eanuts) 4 and 9
11. KilledBy(Bill, P eanuts) 5 and 10
12. ⊥ 6 and 11

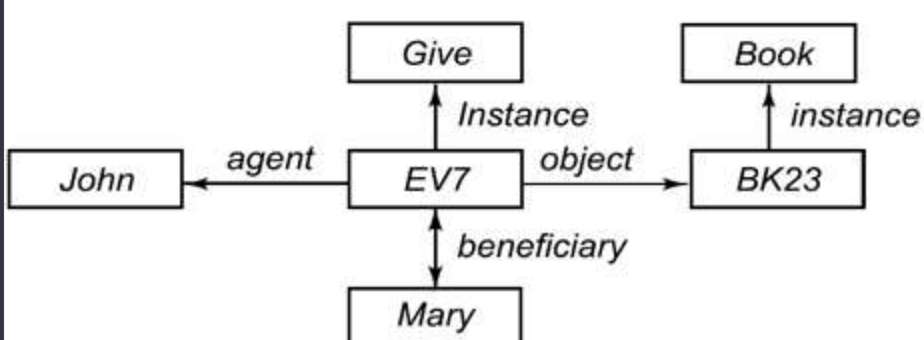
•

(e) Use resolution to answer the question, "What food does Sue eat?"

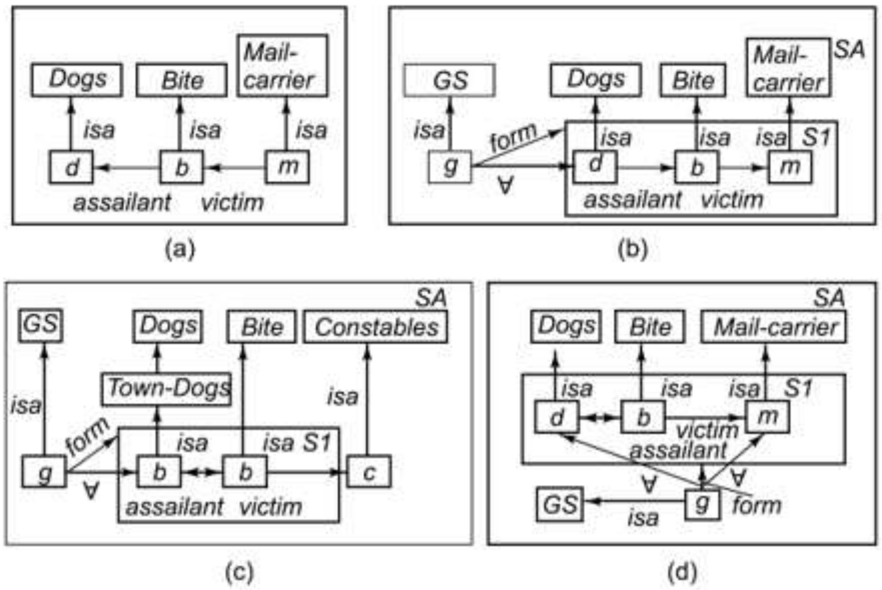
13. ¬Eats(Sue, x) negated query
14. ¬Eats(Bill, x) 7 and 13
15. ⊥ 5 and 14, unifying P eanuts/x

A Semantic Net Representing a Sentence

- "John gave the book to Mary."



Partitioned Semantic Nets



a) The dog bit the mail carrier.
b) Every dog has bitten a mail carrier.
c) Every dog in town has bitten the constable.
d) Every dog has bitten every mail carrier.

Bayesian Network with Conditional Probability Matrix:

makefile Copy code

```makefile
Variables:
S = Patient has spots
M = Patient has measles
F = Patient has high fever
V = Patient has previously been vaccinated against measles
T = Patient was recently bitten by tick
A = Patient has an allergy

Conditional Probability Matrix:
P(S|M) = High (Measles causes spots)
P(F|M) = High (Measles causes fever)
P(M|V) = Low (Vaccination reduces likelihood of measles)
P(A|T) = High (Tick bite increases likelihood of allergy)
P(S|A) = Moderate (Allergy could cause spots)
P(F|A) = Low (Allergy might exacerbate fever)
```

Nodes:

    - Patient has spots
    - Patient has measles
    - Patient has high fever
    - Patient has previously been vaccinated against measles
    - Patient was recently bitten by tick
    - Patient has an allergy
- Causal Connections:
    - Measles vaccination influences the likelihood of having measles.
    - Measles causes spots and high fever.
    - Being bitten by a tick might cause an allergy.
    - Allergy could cause spots or could exacerbate measles symptoms.

Sure, let's create two fuzzy relations, R and S, represented by matrices, and then find their max-min and max-product compositions.

Let's consider two fuzzy relations R and S as:

lua

```lua
R = [[0.8, 0.4],
     [0.6, 0.3]]

S = [[0.7, 0.2],
     [0.5, 0.6]]
```

Max-Min Composition:

For the max-min composition of R and S, we'll perform the following operations:

For element (i, j) in the resulting matrix C:

- C(i, j) = max(min(R(i, k), S(k, j))) for k in the range of the matrix size.

Let's calculate:

scss

```scss
C_max-min(1, 1) = max(min(0.8, 0.7), min(0.4, 0.2)) = max(0.7, 0.2) = 0
C_max-min(1, 2) = max(min(0.8, 0.2), min(0.4, 0.6)) = max(0.2, 0.4) = 0
C_max-min(2, 1) = max(min(0.6, 0.7), min(0.3, 0.2)) = max(0.6, 0.2) = 0
C_max-min(2, 2) = max(min(0.6, 0.2), min(0.3, 0.6)) = max(0.2, 0.3) = 0

C_max-min = [[0.7, 0.4],
             [0.6, 0.3]]
```

Max-Product Composition:

For the max-product composition of R and S, we'll perform the following operations:

For element (i, j) in the resulting matrix D:

- D(i, j) = max(R(i, k) * S(k, j)) for k in the range of the matrix size.

Let's calculate:

```scss
D_max-product(1, 1) = max(0.8 * 0.7, 0.4 * 0.5) = max(0.56, 0.2) = 0.56
D_max-product(1, 2) = max(0.8 * 0.2, 0.4 * 0.6) = max(0.16, 0.24) = 0.2
D_max-product(2, 1) = max(0.6 * 0.7, 0.3 * 0.5) = max(0.42, 0.15) = 0.4
D_max-product(2, 2) = max(0.6 * 0.2, 0.3 * 0.6) = max(0.12, 0.18) = 0.1

D_max-product = [[0.56, 0.24],
                 [0.42, 0.18]]
```