UNIT – I: Introduction

Concept of AI, History, Current status, Scope, Agents, Environments, Problem Formulations, Review of tree and graph structures, State space representation, Search graph and Search tree.

Concept of AI

What is AI ?

The answer to this question would depend on who you ask. A layman, with a fleeting understanding of technology, would link it to robots. If you ask about artificial intelligence to an AI researcher, (s)he would say that it's a set of algorithms that can produce results without having to be explicitly instructed to do so. Both of these answers are right. So to summarize, Artificial Intelligence is:

- An intelligent entity created by humans.
- Capable of performing tasks intelligently without being explicitly instructed.
- Capable of thinking and acting rationally and humanely.

Current Status /Applications of AI

- **Google Smart compose**
- **Google Translate**
- **Smart Email Categorization**- categorize your emails into primary, social, and promotion inboxes, as well as labeling emails as important. Every time you mark an email as important, Gmail learns.
- Email spam filters
- Voice-to-text and Speech recognition
- Auto complete
- Personliazed ads
- Google's AI-powered predictions (E.g.: Google Maps)

- Ride-sharing applications (E.g.: Uber, Lyft)
- AI Autopilot in Commercial Flights
- Plagiarism checkers and tools
- Facial Recognition
- Search recommendations
- Next word prediction, spell and Grammar check
- **Virtual assistants** (Alexa, Cortona, Siri, Google Assistant, Amazons Echo)- even controls their appliances at home, It can do more than just play your favorite songs. It can be used to control the devices at your house, book cabs, make phone calls, order your favorite food, check the weather conditions, and so on.
- **Chat bots**-Banking=> AI in banking is growing faster than you thought! A lot of banks have already adopted AI-based systems to provide customer support, detect anomalies and credit card frauds. An example of this is HDFC Bank. HDFC Bank has developed an AI-based chatbot called *EVA* (Electronic Virtual Assistant), built by Bengaluru-based Senseforth AI Research. Since its launch, Eva has addressed over 3 million customer queries, interacted with over half a million unique users, and held over a million conversations. Eva **can collect knowledge from thousands of sources and provide simple answers in less than 0.4 seconds.**
- **Finance**=> Stock market trading Prediction (Nomura-investment management)
- **Marketing**=> Personalized Recommendation Systems using search/watched history and profile meta data (friend in FB or products/movies in NETFLIX/AMAZON)- when we search for an item on any e-commerce store, we get all possible results related to the item. It's like these search engines are reading our minds! In a matter of seconds, we get a list of all relevant items. An example of this is finding the right movies on Netflix.
- Auto face detection and tagging of faces in facebook
- Humanoids (Sophia)
- Google's Self Driving cars (Tesla autopilot feature)
- Image captioning (explain content in image)
- Auto Summarization (text, video-multi view/modal, scientific articles)
- Sentiment Analysis
- **Smart homes**- smart doors, switches, AC control (auto heat and cool) compatible with mobile devices
- **Agriculture** =>Organizations are using automation and robotics to help farmers find more efficient ways to protect their crops from weeds. **Blue River Technology has developed a robot called See & Spray** which uses computer vision technologies like *object detection* to monitor and precisely spray weedicide on cotton plants. Precision spraying can help prevent herbicide resistance. Apart from this, Berlin-based agricultural tech start-up called PEAT, has developed an application called **Plantix that identifies potential defects and nutrient deficiencies in the soil through images.**The image recognition app identifies possible defects through images captured by the user's smartphone camera. Users are then provided with **soil**

restoration techniques, tips, and other possible solutions. The company claims that its software can achieve pattern detection with an estimated accuracy of up to 95%.

- **Health care** =>An organization called Cambio Health Care developed a clinical decision support system for stroke prevention that can **give the physician a warning** when there's a patient at risk of having a heart stroke. Another such example is Coala life which is a company that has a digitalized device that can find cardiac diseases. Similarly, Aifloo is developing a system for keeping track of how people are doing in nursing homes, home care, etc. The best thing about AI in healthcare is that you don't even need to develop a new medication. Just by using an existing medication in the right way, you can also save lives.
- **AI in education:**
  AI can **automate grading** so that the tutor can have more time to teach. AI chatbot can **communicate with students** as a teaching assistant.
  AI in the fture can be work as a **personal virtual tutor** for students, which will be accessible easily at any time and any place.
- **AI in Social Media**
  Social Media sites such as Facebook, Twitter, and Snapchat Filters contain billions of user profiles, which need to be stored and managed in a very efficient way. AI can organize and manage massive amounts of data. AI can analyze lots of data to identify the latest trends, hashtag, and requirement of different users.
  Facebook uses Machine Learning algorithms to suggest friends to us with its 'People You May Know' feature. Also, it uses Machine Learning for face detection, i.e., recognizing faces in a group photo.
- **AI in Travel & Transport**
  AI is becoming highly demanding for travel industries. AI is capable of doing various travel related works such as from making travel arrangement to suggesting the hotels, flights, and best routes to the customers. Travel industries are using AI-powered chatbots which can make human-like interaction with customers for better and fast response.

Definition of AI

"It is a branch of computer science by which we can create intelligent machines which can behave like a human, think like humans, and able to make decisions."

(without being explicitly pre-programmed, the system has to work with its own intelligence (man-made thinking power).

- Some definitions measure success in terms of human performance, whereas some measure against ideal concept of intelligence, that is, rationality. A system is rational if it does right things.
- This gives four possible goals to peruse AI. So definitions are organized into four categories with **two dimensions:**
  1) Thinking versus Acting
  2) Human versus Rational

1. **Systems that think like humans:** The exciting new effort to make computers as thinking machines with brain.

2. **Systems that think rationally:** The study of the computations that make it possible to perceive, reason, and act correctly.
3. **Systems that act like humans:** The study of how to make computers do things at which, at the moment, people are better.
4. **Systems that act rationally**: The branch of computer science that is concerned with the automation of intelligent behaviour.

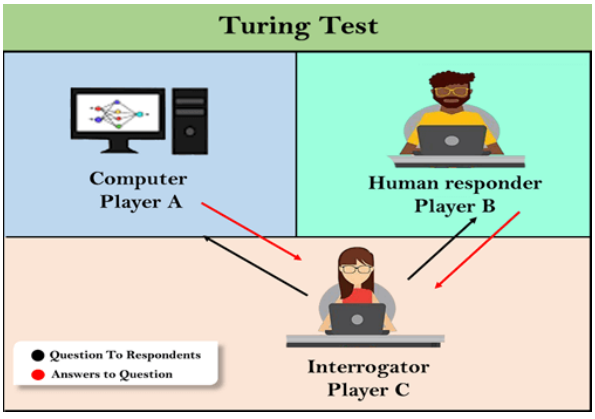
	Human-like Intelligence	"Ideal" Intelligent/ Pure Rationality	
Thought/ Reasoning	2. Thinking humanly	3. Thinking Rationally	Which is closest to a 'real' human?
Behavior/ Actions "behaviorism"	1. Acting Humanly	4. Acting Rationally	Furthest?

Acting humanly: The Turing Test approach

- **The Turing Test,** proposed by Alan Turing (1950), was designed to provide a satisfactory operational definition of intelligence.
- Turing defined intelligent behaviour as the ability to achieve human- level performance in all cognitive tasks, sufficient to fool an interrogator.

Usually the term "cognitive task(consicousnss)" is used in the context of "cognitive task analysis." This is the process of analyzing or testing what level of thought process has reached or is using to solve a specific problem. You might look at a toddler, for example, who touches a hot iron and gets burned. Examples selecting the "odd man out" among three or more alternatives, identifying whether a single presented number (or letter) was or was not part of a previously presented set,

- The test he proposed is that the computer should be interrogated by a human via a teletype, and passes the test if the interrogator cannot differentiate responses between a computer or a human at the other end.



□ □ The computer would need to possess the following capabilities:

- **natural language processing** helps computers to understand the way that humans write and speak and enables it to communicate successfully with an intelligent system using a natural language like English (or some other human

language) and making the machine to understand, analyse, manipulate, and interpret human's languages.

- **knowledge representation** to store information provided before or during the interrogation; It is responsible for representing information about the real world so that a computer can **understand and can utilize this knowledge to solve the complex real world problems** such as diagnosing a medical condition or communicating with humans in natural language. Knowledge representation is not just storing data into some database, but it also enables an intelligent machine to **learn from that knowledge and experiences** so that it can behave intelligently like a human.
- **automated reasoning**,Reasoning plays a great role in the process of artificial Intelligence. Thus Reasoning can be defined as the logical process of **drawing conclusions, making predictions** or constructing approaches towards a particular thought with the **help of existing knowledge**. In artificial intelligence, reasoning is very important because to understand the human brain, how the brain thinks; how it draws conclusions towards particular things for all these sorts of works we need the help of reasoning.
- **machine learning** is the study of computer algorithms that can improve automatically through experience and by the use of existing data and to adapt to new circumstances and to detect and derive patterns.

 Turing's test deliberately avoided direct physical interaction between the interrogator and the computer, because physical simulation of a person is unnecessary for intelligence.

Thinking humanly: The cognitive modelling approach

- cognitive computing is typically used to describe **AI systems that aim to simulate human thought**.
- Thinking humanly means trying to understand and model how the human mind works.

• There are (at least) two possible routes that humans use to find the answer to a question:

– We reason about it to find the answer. This is called "**introspection".**

– We conduct **psychological experiments** to find the answer, drawing upon scientific techniques to conduct controlled experiments and measure change. To say that a given program thinks like a human, we must have some **way of determining how humans think**. We need to **get inside** the actual workings of human minds to see how it works and then comparing our computer programs to this. This is what cognitive science attempts to do.

- The field of Cognitive Science focuses on modeling how people think, **computer models + experimental techniques from psychology**

Once we have a sufficiently precise theory of the mind, it becomes possible to express the theory as a computer program. If the program's input/output and timing behaviour matches human behaviour, that is evidence that some of the program's mechanisms may also be operating in humans.

Thinking rationally: The laws of thought approach

- Trying to understand **how we actually think** is one route to AI.
- But another approach is to model how we should think.
- The "thinking rationally" approach to AI uses symbolic logic (Symbolic logic is an expression of logic by using symbols in the place of natural language using propositions, truth tables,) to **capture the laws of rational thought as symbols that can be manipulated.**

Symbolic logic example:

Propositions: If all mammals feed their babies milk from the mother (A). If all cats feed their babies mother's milk (B). All cats are mammals(C).

The Λ means "and," and the $\Rightarrow$ symbol means "implies."

Conclusion: $A \Lambda B \Rightarrow C$

Explanation: Proposition A and proposition B lead to the conclusion, C. If all mammals feed their babies milk from the mother and all cats feed their babies mother's milk, it implies all cats are mammals.

- Reasoning involves manipulating the symbols according to well-defined rules, kind of like algebra.
- The Greek philosopher Aristotle was one of the first to attempt to codify "right thinking," that is, irrefutable reasoning processes. His famous syllogisms provided patterns for argument structures that always gave correct conclusions given correct premises. For example, "Socrates is a man; all men are mortal; therefore Socrates is mortal." These laws of thought were supposed to govern the operation of the mind, and initiated the field of logic.

- There are two main obstacles to this approach.
  - First, it is difficult to **make informal knowledge and state it in the formal terms required by logical notation,** particularly when there is **uncertainty** in the knowledge.
  - Second, there is a big difference between being able to solve a problem "**in principle" and doing so in practice.**

Acting rationally: The rational agent approach

- Acting rationally means acting so as to achieve one's goals, given one's beliefs.
- An agent is just something that perceives and acts.
- In this approach, AI is viewed as the study and construction of rational agents.
- playing a game will act rationally if it tries to win the game
- In the 'laws of thought" approach to AI, the whole emphasis was on correct inferences. This is often part of being a rational agent because **one way to act rationally is to reason logically and then act on ones conclusion**s.
- On the other hand, correct inference is not all of rationality, because there are often situations where there is **no provably correct thing to do**, yet something must still be done. There are also ways of acting rationally that cannot be reasonably said to involve inference. For example, pulling one's hand off of a hot stove is a reflex action that is more successful than a slower action taken after careful deliberation.

- The study of AI as rational agent design has an advantage that it is more general than the logical approach because correct inference is only a useful mechanism for achieving rationality, not a necessary one.

	Humanly	Rationally
Thinking	Thinking humanly — cognitive modeling. Systems should solve problems the same way humans do.	Thinking rationally — the use of logic. Need to worry about modeling uncertainty and dealing with complexity.
Acting	Acting humanly — the Turing Test approach.	Acting rationally — the study of rational agents: agents that maximize the expected value of their performance measure given what they currently know.

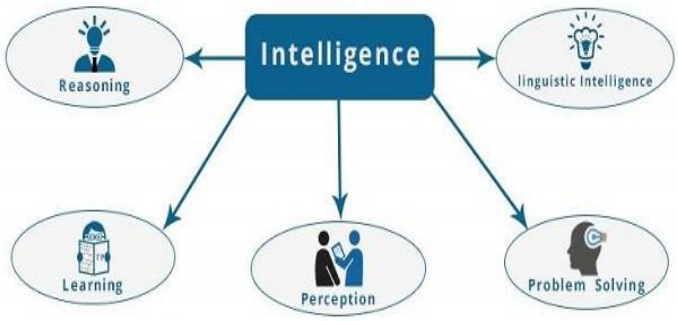
Components of AI

Goal of AI

Replicate/mimic human intelligence, Solve knowledge intense tasks with more accuracy and concise, precision results, is a connection of Perceptions and actions.

Driving car in traffic using sensors and image processing, Playing chess, doing surgical operations like human.

Can exhibit intelligence behaviour, lean new things by itself, demonstrate and explain, advise new things itself to user (why it has taken those decisions logical reasoning behind the decision taken)



Artificial Intelligence exists when a machine can have human based skills such as below

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

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

Learning- Learning is the process of gaining knowledge by studying, understanding, practicing, being taught, or experiencing some thing. Learning enhances the awareness of any topic.

Linguistic intelligence- Linguistics- The ability to speak, recognize, and use mechanisms of phonology (speech sounds), syntax (grammar), and semantics (meaning).
Linguistic intelligence is one's ability to use, comprehend, speak, and written language. It is important in interpersonal communication.

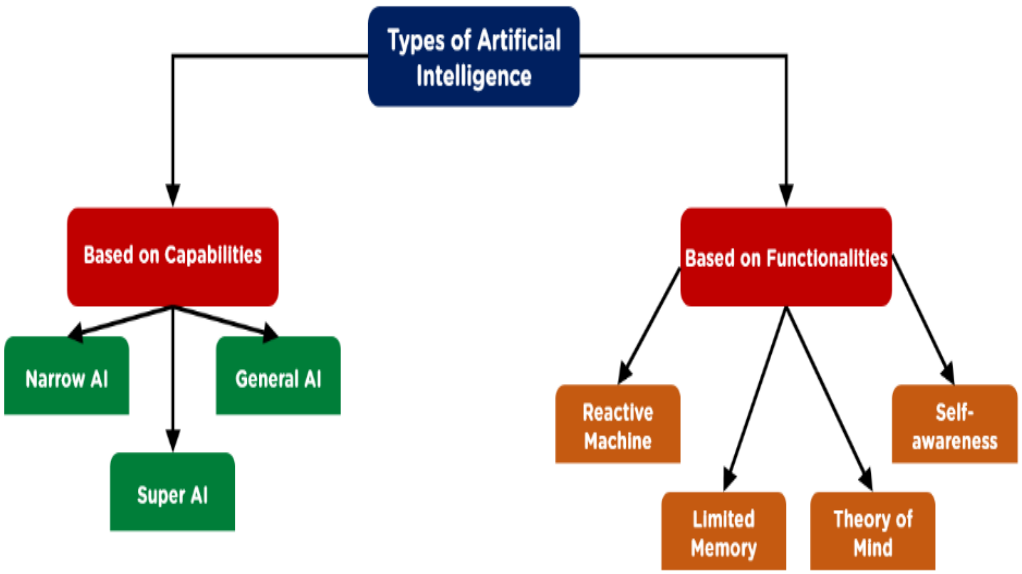
Problem solving- Problem solving is the method during which **one perceives** and tries to make a **desired answer from a present** state of affairs by taking some path, that is blocked by known or unknown hurdles. It is the process of **selecting the best suitable alternative** out of multiple alternatives to reach the desired goal are available.

Reasoning- Reasoning is the set of processes that enables an intelligent system to help or to provide basis for judgement/actions, making decisions, and prediction.
Reasoning is of two types: Inductive Reasoning and Deductive Reasoning.

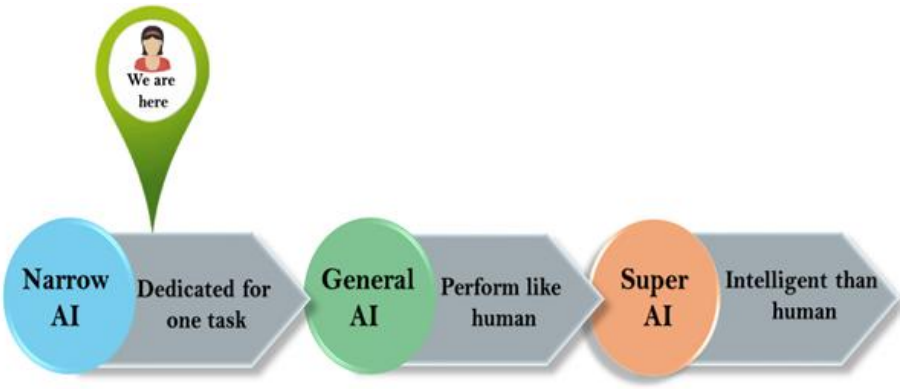
Inductive reasoning help in making generalization from specific facts and knowledge. Inductive reasoning is the bottom-up process. (goal--> steps followed to reach that goal)	It starts with a general statement uses available facts, information or knowledgeand examines the possibilities to draw a specific, logical conclusion. (start with a step1 then use some steps to reach to goal)

Levels /Types of Artificial Intelligence

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



AI type-1: Based on Capabilities



1. Weak AI or Narrow AI

- Narrow AI, also called as Weak AI, focuses on one narrow task designed to **perform dedicated task** and cannot perform beyond its limitations.
- Although these machines are seen to be intelligent, they function under minimal limitations, and thus, are referred to as weak AI. It does not mimic human intelligence; it stimulates human behaviour based on certain parameters.
- Narrow AI makes use of NLP or natural language processing to perform tasks.

Examples

- Siri, Alexa, Cortana (Personal voice assistant operates with a limited pre-defined range of functions.)
- IBMs Watson voice-to-text /Virtual assistant
- Self-driving cars
- google translate
- image recognition software
- spam filtering
- Google's page-ranking algorithm.
- Recommendation systems

If you ask Alexa to turn on the TV, the programming understands key words like **On** and **TV**. The algorithm will respond by turning on the TV, but it is only responding to its programming. In other words, it does not comprehend any of the meaning of what you said.

2. General/ Strong AI:

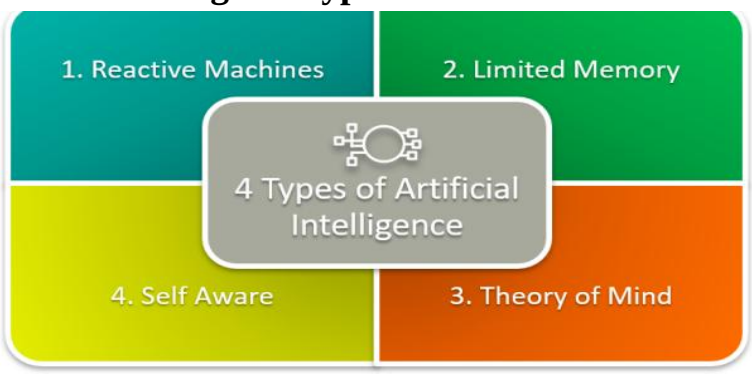
- Also known as strong AI or deep AI, artificial general intelligence refers to the concept through which **machines can mimic human intelligence** while showcasing the **ability to apply their intelligence to solve problems**.
- Scientists have not been able to achieve this level of intelligence yet. Significant research needs to be done before this level of intelligence can be achieved. Scientists would have to find a way through which machines can become conscious through programming a set of cognitive abilities.

3.Super AI:

- Super-intelligence can be known as that level wherein the **machine surpasses human intelligence capabilities and becomes self-aware and having emotions.**It's when machines develop consciousness and decision-making on par (or better) than humans.
- Some key characteristics include the ability to think, to reason, solve the puzzle, make judgments, plan, learn, and communicate by its own.
- This concept has been the muse to several films, and science fiction novels wherein robots who are capable of developing their feelings and emotions can overrun humanity itself.
- It would be able to build emotions of its own, and hypothetically, be better than humans at art, sports, math, science, and more. The decision-making ability of a super-intelligence would be greater than that of a human being.
- Ex: Atari, Alpha-Go Games

Weak AI	**Strong AI**
It is a narrow application with a limited scope.	It is a wider application with a more vast scope.
This application is good at specific tasks.	This application has an incredible human-level intelligence.
It uses supervised and unsupervised learning to process data.	It uses clustering and association to process data.
Example: Siri, Alexa.	Example: Advanced Robotics

Artificial Intelligence type-2: Based on functionality



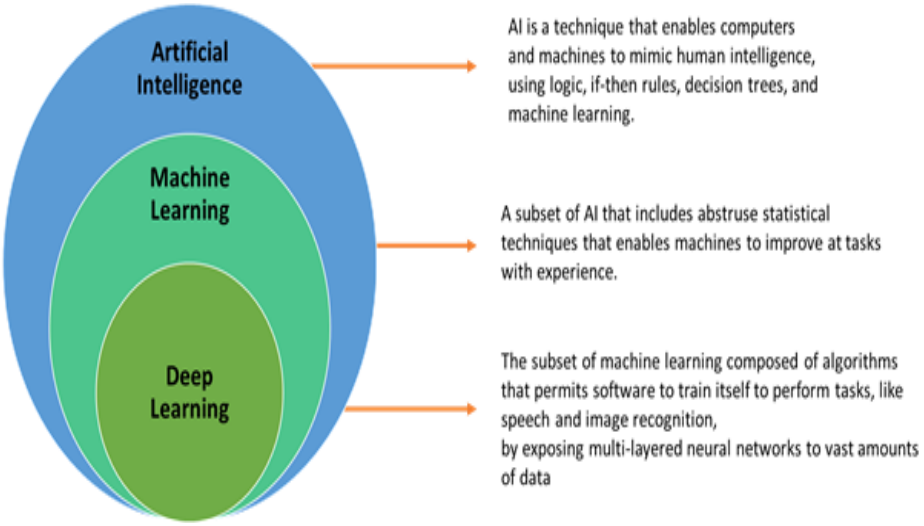
1. **Reactive Machines**
   - Purely reactive machines are the most basic types of Artificial Intelligence.
   - Such AI systems **do not store memories or past experiences for future actions.**
   - These machines **considers only current scenarios and react on it as per possible best action**.They perceive the world and react to it.
   - Examples of Reactive AI is the famous **IBM Deep Blue system Chess program that beat the world champion**, Garry Kasparov, Deep Blue can make predictions about what moves might be next for it and its opponent. It ignores everything before the present moment and looks at the chessboard pieces as it stands right now and chooses from possible next moves and one more example is Google's AlphaGo

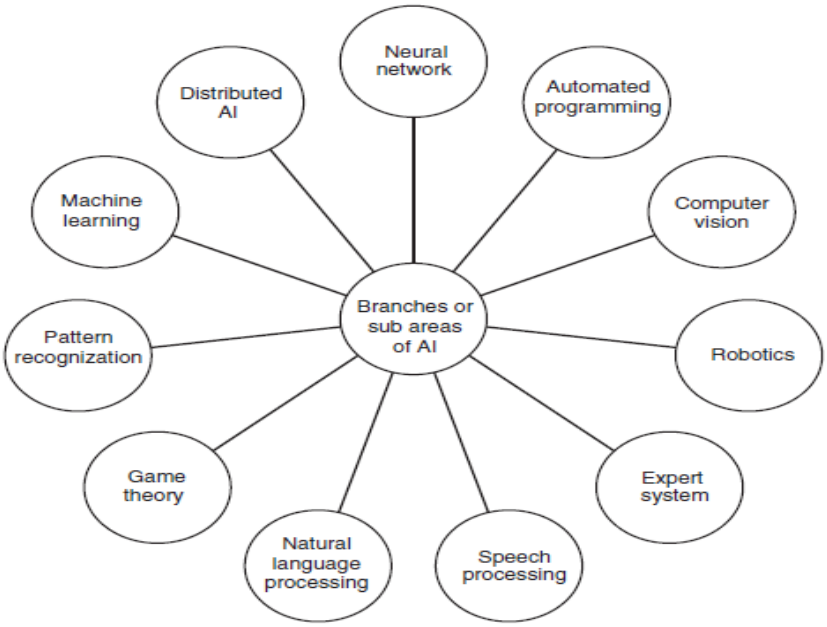
2.**Limited Memory**

- Like the name suggests Limited Memory AI, can make informed and **improved decisions by studying the past data from its memor**y. Such an AI has a short-lived or a **temporary memory t**hat can be used to store past experiences and hence evaluate future actions, but they cannot add it to a library of their experiences.
- Self-driving cars are one of the best examples of Limited Memory systems. These cars can store recent speed of nearby cars, the distance of other cars, speed limit, and other information to navigate the road and uses sensors to identify civilians crossing the road, steep roads, traffic signals and so on to make better driving decisions. This helps to prevent any future accidents.

3. **Theory of Mind**
   - This category of machines is speculated to play a major role in psychology.
   - This kind of AI requires a thorough understanding that the people and things within an environment can alter feelings and behaviors. Theory of Mind AI should **understand people's emotions, sentiments, and thoughts and be able to interact socially like humans.**
   - Even though many improvements are there in this field, this kind of AI is not fully complete yet.
   - One real-world example of the theory of mind AI is Kismet. **Kismet is a robot head** can mimic human emotions and recognize them. Both abilities are key advancements in theory of mind AI, but Kismet can't follow gazes or convey attention to humans.
   - **Sophia** from Hanson Robotics is another example where the theory of mind AI was implemented. Cameras present in Sophia's eyes, combined with computer algorithms, allow her to see. She can sustain eye contact, recognize individuals, and follow faces, have emotions and feelings.
4. **Self-Awareness**
   - Self-Awareness AI does not exist in reality still and it is a hypothetical concept.
   - Such systems **understand their internal traits, states, and conditions and perceive human emotions. These machines will be super intelligent, and will have their own consciousness, sentiments, and self-awareness.** This type of AI will not only be able to understand and evoke emotions in those it interacts with, but also have emotions, needs, and beliefs of its own.
   - Self-awareness AI is the future of Artificial Intelligence. These machines will be smarter than human mind.

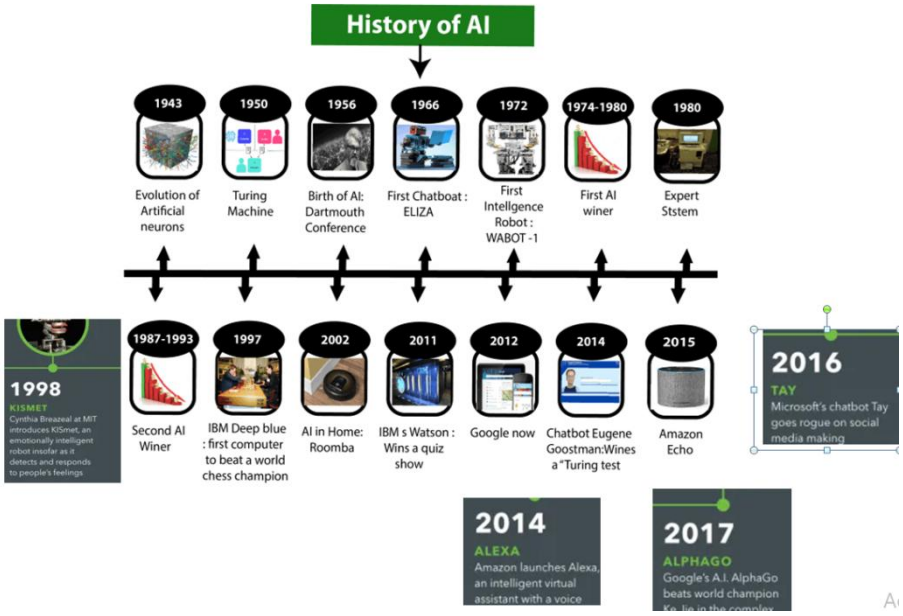
Sub-fields of AI





History of Artificial Intelligence

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



Maturation of Artificial Intelligence (1943-1952)

- **Year 1943:** The first work which is now recognized as AI was done by Warren McCulloch and Walter pits in 1943. They proposed a model of **artificial neurons**.
- **Year 1949:** Donald Hebb demonstrated an updating rule for modifying the connection strength between neurons. His rule is now called **Hebbian learning**.
- **Year 1950:** The Alan Turing who was an English mathematician and pioneered Machine learning in 1950. Alan Turing publishes **"Computing Machinery and Intelligence"** in which he proposed a test. The test can check the machine's ability to exhibit intelligent behavior equivalent to human intelligence, called a **Turing test**.

The birth of Artificial Intelligence (1952-1956)

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

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

The golden years-Early enthusiasm (1956-1974)

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

The first AI winter (1974-1980)

- The duration between years 1974 to 1980 was the first AI winter duration. AI winter refers to the time period where computer scientist dealt with a severe shortage of funding from government for AI researches.

- During AI winters, an interest of publicity on artificial intelligence was decreased.

A boom of AI (1980-1987)

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

The second AI winter (1987-1993)

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

The emergence of intelligent agents (1993-2011)

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

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

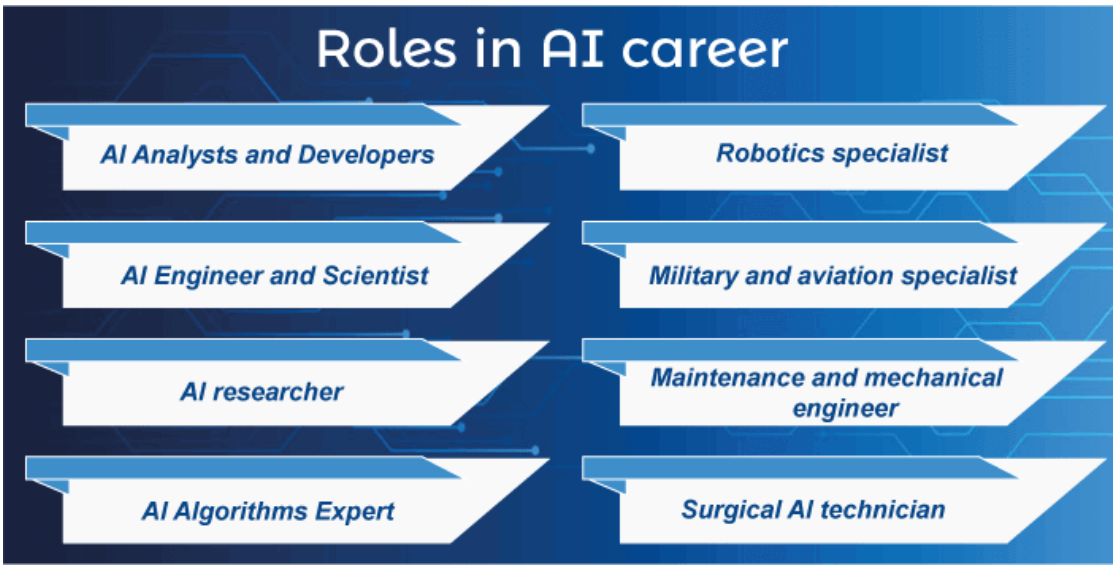
- **Year 2011:** In the year 2011, IBM's Watson won jeopardy, a quiz show, where it had to solve the complex questions as well as riddles. Watson had proved that it could understand natural language and can solve tricky questions quickly.
- **Year 2012:** Google has launched an Android app feature "Google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

Now AI has developed to a remarkable level. The concept of Deep learning, big data, and data science are now trending like a boom. Nowadays companies like Google, Facebook, IBM, and Amazon are working with AI and creating amazing devices. The future of Artificial Intelligence is inspiring and will come with high intelligence.

Year	Milestone / Innovation
1923	KarelČapek plays named "Rossum's Universal Robots, the first use of the word "robot" in English.
1943	Foundations for neural networks laid.
1945	Isaac Asimov, a Columbia University alumni, use the term Robotics.
1956	John McCarthy first used the term Artificial Intelligence. Demonstration of the first running AI program at Carnegie Mellon University.
1964	Danny Bobrow's dissertation at MIT showed how computers could understand natural language.
1969	Scientists at Stanford Research Institute Developed Shakey. A robot equipped with locomotion and problem-solving.
1979	The world's first computer-controlled autonomous vehicle, Stanford Cart, was built.
1990	Significant demonstrations in machine learning
1997	The Deep Blue Chess Program beat the then world chess champion, Garry Kasparov.
2000	Interactive robot pets have become commercially available. MIT displays Kismet, a robot with a face that expresses emotions.
2006	AI came into the Business world in the year 2006. Companies like Facebook, Netflix, Twitter started using AI.
2012	Google has launched an Android app feature called "Google now", which provides the user with a prediction.
2018	The "Project Debater" from IBM debated complex topics with two master debaters and performed exceptionally well.

Scope / Future of Artificial Intelligence (AI careers)

Artificial intelligence is the next big revolution which is changing the world. As per " Forbes " the world is getting more intelligent day by day. The estimate is that the majority of the tasks will be performed by AI and machines by the year 2030. Artificial intelligence is a very powerful technology which is changing the world. It is one of the most exciting and promising technologies of the 21st century. Artificial intelligence is affecting various industries including transport, healthcare, ecommerce, etc. It is changing the world in so many ways. It is affecting each and every domain of life.



1. AI in Science and Research

AI is making lots of progress in the scientific sector. Artificial Intelligence can handle large quantities of data and processes it quicker than human minds. This makes it perfect for research where the sources contain high data volumes.

AI is already making breakthroughs in this field. A great example is 'Eve,' which is an AI-based robot. It discovered an ingredient of toothpaste that can cure a dangerous disease like Malaria. Imagine a common substance present in an everyday item that is capable of treating Malaria; it's a significant breakthrough, no doubt.

Drug discovery is a fast-growing sector, and AI is aiding the researchers considerably in this regard. Biotechnology is another field where researchers are using AI to design microorganisms for industrial applications.

2. AI in Cyber Security

Cybersecurity is another field that's benefitting from AI. As organizations are transferring their data to IT networks and cloud, the threat of hackers is becoming more significant.

One triumphant attack can wreak havoc on an organization. To keep their data and resources secure, organizations are making massive investments in cybersecurity. The future scope of AI in cybersecurity is bright.

Cognitive AI is an excellent example of this field. It detects and analyses threats, while also providing insights to the analysts for making better-informed decisions.

Many institutions are using AI-based solutions to automate the repetitive processes present in cybersecurity. For example, IBM has IBM Resilient, which is an agnostic and open platform that gives infrastructure and hub for managing security responses.

Another field is fraud detection. AI can help in detecting frauds and help organizations and people in avoiding scams. They can scan extensive quantities of transactions quickly and classify them according to their trustworthiness. By identifying fraudulent transactions and tendencies, organizations can save a lot of time and resources. It surely lessens the risk of losing money.

3. AI in Data Analysis

AI algorithms are capable of improving with iterations, and this way, their accuracy, and precision increase accordingly. AI can help data analysts with handling and processing large datasets.

AI can identify patterns and insights that human eyes can't notice without putting in a lot of effort.

As mentioned earlier, AI systems can handle tons of data and process it much faster than humans. So, they can take customer data and make more accurate predictions of customer behavior, preferences, and other required factors. Helixa.ai is a great example of such an AI application. They use AI to provide insights into customer (or audience) behavior for higher accuracy and better results. Agencies and marketers can use their services to build precise buyer personas and create better-targeted ad campaigns.

4. AI in Transport

AI & ML sees a tremendous scope in the transportation sector as well. In ships, aircraft, spacecraft; autopilot has been in use since 1922. The transport sector has been using AI for decades. Airplanes have been using autopilot to steer them in the air.

Ships and spacecraft also use autopilot to help them maintain the correct course.

Autopilot helps the human operator and assists them in heading in the right direction. This allows the pilots to focus on other more important areas of the flight, such as the weather and the trajectory of the plane.

Another area where the future scope of AI is quite broad is driverless cars. Many companies are developing autonomous vehicles, which will rely heavily on AI and ML to operate optimally. Experts believe self-driving cars will bring many long-term and short-term benefits, including lower emissions and enhanced road safety. For example, self-driving cars will be free from human errors, which account for 90% of traffic accidents. Many companies, including Tesla and Uber, are developing these vehicles.

5. AI in Home

AI has found a special place in people's homes in the form of Smart Home Assistants. Amazon Echo and Google Home are popular smart home devices that let you perform various tasks with just voice commands.

You can order groceries, play music, or even switch on/off the lights in your living room with just a few voice commands. Both of them rely on Voice Recognition technologies, which are a result of Artificial Intelligence and Machine Learning. They constantly learn from the commands of their users to understand them better and become more efficient.

Smart assistants are also present in mobile phones. Apple's Siri and Google Assistant are great examples of this sort. They also learn to recognize their users' voices to interpret them better all the time. And they can perform a plethora of tasks. Microsoft also has a smart assistant, which is called Cortana.

You can use these smart assistants for various tasks such as:

- Playing a song
- Asking a question
- Buying something online
- Opening an app

There's a lot of room left for improvement, but surely, the scope of AI in the smart home sector is booming.

6. AI in Healthcare

The medical sector is also using this technology for its advantages. AI is helping medical researchers and professionals in numerous ways.

For example, the Knight Career Institute and Intel have made a collaborative cancer cloud. This cloud takes data from the medical history of cancer (and similar) patients to help doctors in making a better diagnosis. Preventing cancer from moving to higher stages is its most effective treatment at this time.

Apart from finding a cure for cancer, some organizations are using AI to help patients get telemedicine. The UK's National Health Service uses Google's DeepMind platform to detect health risks in people through apps.

Wrong diagnoses are a significant problem in the medical sector. AI can help doctors in avoiding these errors by providing them with relevant databases and recommendations. It can analyze the database of patients with similar symptoms and suggest the treatment that was the most successful in those cases.

Many major organizations, including IBM and Microsoft, are collaborating with medical institutions to solve the various problems present in the healthcare sector.

AI can also help in reducing medical costs by preventing diseases beforehand and helping doctors in making better diagnoses. BCIs (Brain-computer Interfaces) is another area where the medical sector is utilizing AI. These interfaces help in predicting problems related to speaking or moving that might develop due to problems in the brain. They use AI to help these patients overcome these issues, too, by decoding neural activates.

5. **Manufacturing industry-**

Several global startups which are serving the manufacturing industry are actually based on artificial Intelligence and machine learning. These companies are now developing AI-based solutions for increasing their revenue. One of the unique techniques of AI in the manufacturing industry is to analyze the data and make the best future predictions.

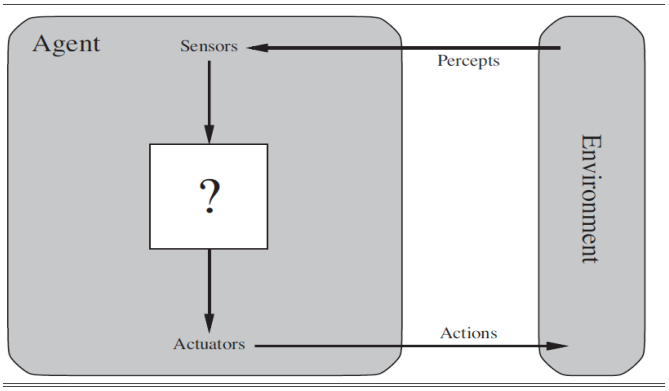
7. **Agriculture-**

The Indian population is mainly dependent on agriculture for their livelihood. And the problem with Indian farmers is that they are greatly dependent on the conventional methods of farming. But thanks to the advent of AI which has made things easy or for them. Artificial intelligence is now used in agriculture for forecasting behavior along with studying the parasites breeding in the crops. This is something very useful as the farmers can out prepare themselves in advance for insect control. Devices like thermal imaging cameras can now keep a track of the quantity of water usage in a particular agricultural land. In this way, AI is helping the agriculture industry in India to grow.

Agents

What is an Agent?

An **agent** is anything that can be viewed as perceiving its **environment** through **sensors** and acting upon that environment through **actuators**. This simple idea is illustrated in Figure below.



Agents interact with environments through sensors and actuators.

An agent can be:

- **Human-Agent:** A human agent has eyes, ears, and other organs which work for sensors and hand, legs, vocal tract and so on for actuators.
- **Robotic Agent:** A robotic agent can have cameras, infrared range finder for sensors and various motors for actuators.

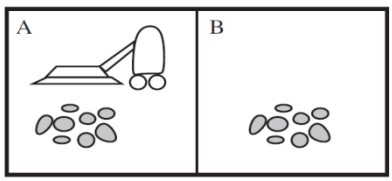
- **Software Agent:** A software agent receives keystrokes, file contents, and network packets as sensory inputs and acts on the environment by displaying on the screen, writing files, and sending network packets.

An Agent runs in the cycle of **perceiving**, **thinking**, and **acting**.

Some terms to know:

- **Perception**: what agent see in the environment thru sensors(sense organs).
  Sensor: Sensor is a device which detects the change in the environment and sends the information to other electronic devices. A robotic/autonomous agent observes its environment through sensors.
- **Perception history**: It is the history of agent perception which comes in a specific period.
- **Actuators:** an actuator is the actual mechanism that enables the effector to execute an action, component of machines that converts energy into motion. The actuators are only responsible for moving and controlling a system. An actuator can be an electric motor, gears, rails, etc.
- **Effectors:** Effectors are the Agents organs (hands and legs)/ a device that affects physical environment. Effectors can be legs, wheels, arms, fingers, wings, fins, and display screen.
- **agent function** that maps any given percept sequence to an action, Agent's behavior is *mathematically* described by **Agent function**
- The agent function for an artificial agent will be implemented practically by an **agent program**.

example—the vacuum-cleaner world shown in Figure below.



A vacuum-cleaner world with just two locations.

This particular world has just two locations: squares A and B.

One very simple **agent function** is the following:

- Percepts: location/which square it is in and contents, e.g., [A,Dirty]
- Actions:Move*Left, Right, Suck up dirt, NoOp*
- *Agent's function* → *look-up table,* if the current square is dirty, then suck; otherwise, move to the other square.*For many agents this is a very large table*

Percept sequence	Action
$[A, Clean]$	$Right$
$[A, Dirty]$	$Suck$
$[B, Clean]$	$Left$
$[B, Dirty]$	$Suck$
$[A, Clean], [A, Clean]$	$Right$
$[A, Clean], [A, Dirty]$	$Suck$
⋮	⋮
$[A, Clean], [A, Clean], [A, Clean]$	$Right$
$[A, Clean], [A, Clean], [A, Dirty]$	$Suck$
⋮	⋮

3 Partial tabulation of a simple agent function for the vacuum-cleaner world

- An **agent program** for this agent is shown in Figure below

```
function REFLEX-VACUUM-AGENT([location,status]) returns an action
  if status = Dirty then return Suck
  else if location = A then return Right
  else if location = B then return Left
```

GOOD BEHAVIOR: THE CONCEPT OF RATIONALITY

Rational Agent

A rational agent is one that does/ performs the right things.

For an AI agent, the rational action is most important because in AI reinforcement learning algorithm, for each best possible action, agent gets the positive reward and for each wrong action, an agent gets a negative reward.AI is about creating rational agents to use for game theory and decision theory for various real-world scenarios.

Rationality

The rationality of an agent is measured by its **performance measure** that evaluates any given **sequence of environment states** and **action sequences**, if sequence leads to **desirable state** then it performed well.

No universal performance measure for all tasks and agents. Consider, for example, the vacuum-cleaner agent we might propose to measure performance by

- theamount of dirt cleaned up in a single eight-hour shift.
- A rational agent can maximize this performance measure by cleaning up the dirt, then dumping it all on the floor, then cleaning it up again, and so on.
- A more suitable performance measure would reward the agent for having a clean floor. For example, one point could be awarded for each clean square at each time step (perhaps with a penalty for electricity consumed and noise generated).

As a **GENERAL RULE**, it is better to design performance measures according to w**hat one actually wants in the environment,** rather than according to how one thinks the agent should behave. E.g., in vacuum-cleaner world, We want the floor clean, no matter how the agent behave

- performance measure of a vacuum-cleaner agent could be amount of dirt cleaned up, amount of time taken, amount of electricity consumed, amount of noise generated, etc.

What is rational at any given time depends on four things

- **Performance measure that defines the success criterion.**
- **Agent prior knowledge of its environment.**
- **Best possible actions that an agent can perform.**
- **The agent's percept sequence to date**

This leads to a **definition of a rational agent=>For each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has.***E.g., an exam, maximize marks, based on the questions on the paper & your knowledge*

THE NATURE OF ENVIRONMENTS

Task environments, which are the "problems" to which rational agents are the "solutions."
Specifying the task environment

PEAS Representation

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

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

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

PEAS for vacuum cleaner	
Performance Measure	cleanness, efficiency: distance travelled to clean, battery life, security
Environment	room, table, wood floor, carpet, different obstacles
Actuators	wheels, different brushes, vacuum extractor
Sensors	camera, dirt detection sensor, cliff sensor, bump sensors, infrared wall sensors

PEAS for self-driving cars:

- Performance measure
  - How can we judge the automated driver?
  - Which factors are considered?
    - getting to the correct destination
    - minimizing fuel consumption
    - minimizing the trip time and/or cost
    - minimizing the violations of traffic laws
    - maximizing the safety and comfort, etc.

- Environment
  - A taxi must deal with a variety of roads
  - Traffic lights, other vehicles, pedestrians, stray animals, road works, police cars, etc.
  - Interact with the customer
- Actuators (for outputs)
  - Control over the accelerator, steering, gear shifting and braking
  - A display to communicate with the customers
- Sensors (for inputs)
  - Detect other vehicles, road situations
  - GPS (Global Positioning System) to know where the taxi is
  - Many more devices are necessary

Agent Type	Performance Measure	Environment	Actuators	Sensors
Taxi driver	Safe, fast, legal, comfortable trip, maximize profits	Roads, other traffic, pedestrians, customers	Steering, accelerator, brake, signal, horn, display	Cameras, sonar, speedometer, GPS, odometer, accelerometer, engine sensors, keyboard

Figure 2.4 PEAS description of the task environment for an automated taxi.

Example of Agents with their PEAS representation

Agent Type	Performance Measure	Environment	Actuators	Sensors
Medical diagnosis system	Healthy patient, reduced costs	Patient, hospital, staff	Display of questions, tests, diagnoses, treatments, referrals	Keyboard entry of symptoms, findings, patient's answers
Satellite image analysis system	Correct image categorization	Downlink from orbiting satellite	Display of scene categorization	Color pixel arrays
Part-picking robot	Percentage of parts in correct bins	Conveyor belt with parts; bins	Jointed arm and hand	Camera, joint angle sensors
Refinery controller	Purity, yield, safety	Refinery, operators	Valves, pumps, heaters, displays	Temperature, pressure, chemical sensors
Interactive English tutor	Student's score on test	Set of students, testing agency	Display of exercises, suggestions, corrections	Keyboard entry

Figure 2.5 Examples of agent types and their PEAS descriptions.

Environment:

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

Properties of task environments

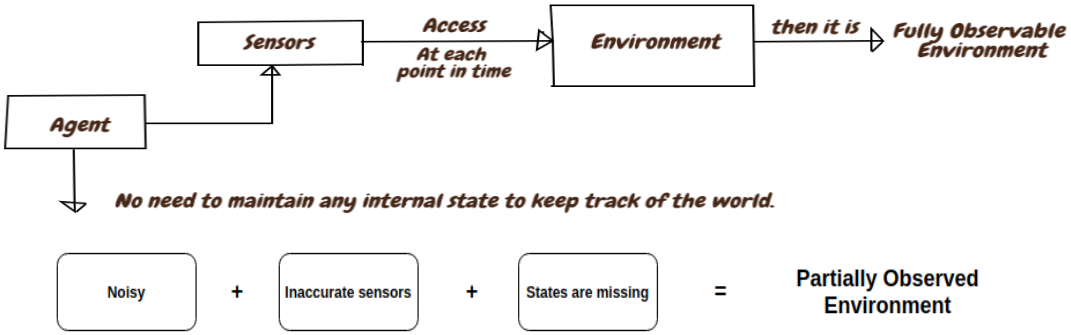
As per Russell and Norvig, an environment can have various features from the point of view of an agent:

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

For example, program a chess bot, the environment is a chessboard and creating a room cleaner robot, the environment is Room**.**

Fully observable vs Partially Observable:

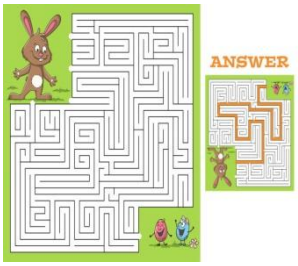
- If an agent sensor can sense or access the complete state of an environment at each point of time then task is **a fully observable** environment, else it is **partially observable**.
- A fully observable environment is easy as there is no need to maintain the internal state to keep track history of the world. Eg. : Sudoku, 8-puzzle problem : The state is completely visible at any point of time.
- **Chess –** the board is fully observable, so are the opponent's moves.
- Eg. : Automated Car Driving System : You cannot predict what other car drivers do, and how they are going to drive or what decision they are going to make while driving.-Partially observable



- An environment is called **unobservable** when the agent has no sensors in all environments.

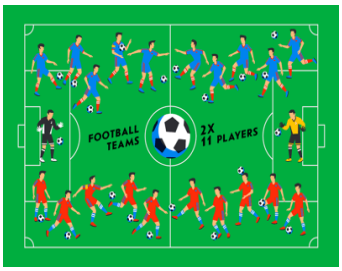
Single-agent vs Multi-agent

- If only one agent is participated/involved in an environment, and operating by itself then such an environment is called single agent environment.
- **Real-life Example**: Playing tennis against the ball is a single agent environment where there is only one player.



- A person left alone in a maze is an example of the single-agent system. A maze is **a type of puzzle games where a player moves in complex and branched passages to find a particular target or location**.
- However, if More than one agent is taking actions in the environment then such an environment is called a multi-agent environment.

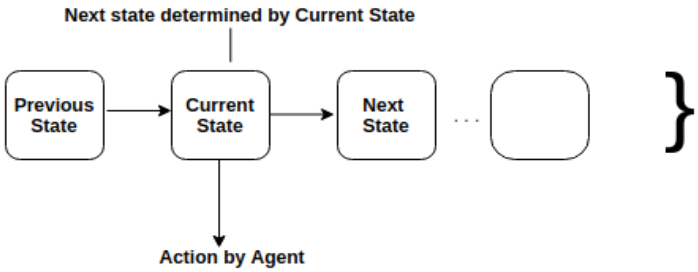
- **Real-life Example**: The game of football is multi-agent as it involves 11 players in each team.



Further, there are two types of "Multi Agent" environments, **Competitive and Collaborative** In competitive Al environments various Al agents are made to **work against** each other with an intent to maximize the outcome. This means that the agents compete with each other to move towards the goal EX, agents compete against each other in order to win the **game of chess**. Whereas, in collaborative AI environments different Al agents **work together** in order to achieve a goal, ex **automated cars, or smart home sensors** work in collaborative Al environments as they work with each other in order to avoid collisions and reach their desired destinations.

Deterministic vs Stochastic:

- When the **next state of the environment is completely determined by the current state & the action executed by the agent,** then we say the environment is deterministic, otherwise it is stochastic.

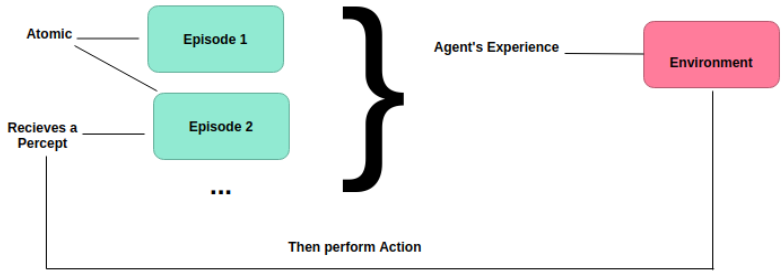


- **Chess –** there would be only a few possible moves for a coin at the current state and these moves can be determined.
- A stochastic environment is random in nature and cannot be determined completely by an agent.

- **Self-Driving Cars –** the actions of a self-driving car are not unique, it varies time to time.

Episodic vs Sequential

- An episode = agent's single pair of perception & action
- In an "Episodic" environment, agent's actions are divided into atomic episodes.
- In each episode, the agent receives a percept and performs a single action. The current episode is different from the previous one and there is no dependency between them. Every episode is independent of each other, Episodic environment is simpler.
- The agent does not need to think ahead.
- **Example:** Consider an example of **Pick and Place robot**, which is used to detect defective parts from conveyer belt. Here, every time robot(agent) will make decision on current part i.e. there is no dependency between current and previous decision.



- **The sequential** environment is where the **next state is dependent on the current action.** So agent current action can change all of the future states of the environment. Here, an agent requires memory of past actions to determine the next best actions.
- Eg. : **Automatic car driving:** If the driver wants to apply break, the next actions include press clutch and then lower the gear.
- **Ex: Checkers-** Where previous move can affect all the following moves.

Static vs Dynamic

- The environment is referred to as "Static", if the **environment remains unchanged while the agent is performing some task/percepting.Static** environments are easy to handle because the agent need not keep looking at the world while deciding on an action, nor need it worry about the passage of time.
- Eg. : Sudoku cross-word puzzle, Vacuum cleaner world : There is no change in the environment when an agent plays sudoku or cleans the dirt, room is static while cleaning, in respective examples.

- **Dynamic** Environment could be changed while an agent is percepting the environment. So agents keep looking at the environment while taking each action.
- Eg : Automatic car driver : The car driver's movement will cause the other car's to make adjustments, thus causing a change in the environment.

- A roller coaster ride is dynamic as it is set in motion and the environment keeps changing every instant.
- Playing soccer is a dynamic environment where players' positions keep changing throughout the game. So a player hit the ball by observing the opposite team.
- A "Semi-Dynamic" environment is the one where environment remains unchanged but the agent's performance changes.
- **Real-life Example**: chess when played with a clock is semi-dynamic,

Discrete vs Continuous

- The types of environments in AI in which we have a **distinct, limited/finite states** and clearly defined input (percepts) and outputs (actions) is called as "Discrete" environment.
- **Real-life Example**: The game of chess is discrete as it has only a finite number of moves. The number of moves might vary with every game, but still, it's finite.

- While in a **Continuous environment**, the environment can have **an infinite number of states.** So the possibilities of taking an action are also infinite cannot be defined beforehand and agent receives a continuous input.
- Eg. : Vision systems in drones or self-driving cars operate on continuous AI environments as their actions are driving, parking, etc. which cannot be finite, env keeps on changing.
- **Real-life Example**: In a basketball game, the position of players (**Environment**) keeps changing continuously and hitting (**Action**) the ball towards the basket can have different angles and speed so infinite possibilities.

Known vs Unknown env

- Known and unknown are not actually a feature of an environment, but it is an agent's state of knowledge to perform an action.
- In a known environment, the results/outcomes/**output for all actions are known**/given to the agent. (example: solitaire card games).
- If the environment is unknown, the agent will **have to learn how it works and has to gain knowledge about how the environment** works for better decision making to perform an action.(example: new video game).

These environments are the good examples of **Exploitation** (Known Environment) and **Exploration** (Unknown Environment) which come in **Reinforcement Learning**.

- There are mainly six groups of environment and an environment can be in multiple groups. Below are more real-life examples and categories into environment groups.

Task Environment	Observable	Agents	Deterministic	Episodic	Static	Discrete
Crossword puzzle	Fully	Single	Deterministic	Sequential	Static	Discrete
Chess with a clock	Fully	Multi	Deterministic	Sequential	Semi	Discrete
Poker	Partially	Multi	Stochastic	Sequential	Static	Discrete
Backgammon	Fully	Multi	Stochastic	Sequential	Static	Discrete
Taxi driving	Partially	Multi	Stochastic	Sequential	Dynamic	Continuous
Medical diagnosis	Partially	Single	Stochastic	Sequential	Dynamic	Continuous
Image analysis	Fully	Single	Deterministic	Episodic	Semi	Continuous
Part-picking robot	Partially	Single	Stochastic	Episodic	Dynamic	Continuous
Refinery controller	Partially	Single	Stochastic	Sequential	Dynamic	Continuous
Interactive English tutor	Partially	Multi	Stochastic	Sequential	Dynamic	Discrete

Figure 2.6 Examples of task environments and their characteristics.

	Fully vs Partially Observable	**Deterministic vs Stochastic**	**Episodic vs Sequential**	**Static vs Dynamic**	**Discrete vs Continuous**	**Single vs Multi Agents**
Brushing Your Teeth	Fully	Stochastic	Sequential	Static	Continuous	Single
Playing Chess	Partially	Stochastic	Sequential	Dynamic	Continuous	Multi-Agent
Playing Cards	Partially	Stochastic	Sequential	Dynamic	Continuous	Multi-Agent
Playing	Partially	Stochastic	Sequential	Dynamic	Continuous	Multi Agent
Autonomous Vehicles	Fully	Stochastic	Sequential	Dynamic	Continuous	Multi-Agent
Order in Restaurant	Fully	Deterministic	Episodic	Static	Discrete	Single Agent

THE STRUCTURE OF AGENTS

Operations of an Agent: Agent Program, Agent Function are the operations.

The task of AI is to design an agent program which implements the agent function.

The structure of an intelligent agent is a combination of architecture and agent program. It can be viewed as: **Agent = Architecture + Agent program**

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

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

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

Agent program: Agent program is an implementation of agent function. An agent program executes on the physical architecture to produce function f. It runs on some sort of computing device with physical sensors and actuators — call this as the Architecture.

Table-lookup agent

• A trivial agent program: keeps track of the percept sequence and then uses it to index into a table of actions to decide what to do

• The designers must construct the table that contains the appropriate action for every possible percept sequence.

```
function TABLE-DRIVEN-AGENT(percept) returns an action
    static:  percepts, a sequence, initially empty
             table, a table of actions, indexed by percept sequences, initially fully specified
    append percept to the end of percepts
    action <--LOOKUP(percepts, table)
    return action
```

Drawbacks:

– Huge table (P^T, **P**: set of possible percepts, **T**: lifetime)

- •Space to store the table
- •Take a long time to build the table
- •No autonomy
- •Even with learning, need a long time to learn the table entries

Rather than a table how we can produce rational behavior from a small amount of code =>

Categorized into five types in **order of increasing generality** a**nd based on their degree of perceived intelligence and capability.** All these agents can improve their performance and generate better action over the time. These are given below:

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

1. Simple Reflex agent:

- The simplest kind of agent is the **simple reflex agent**. These agents **select actions on the basis of the *current* percept, ignoring the rest of the percept history.**
- w/o any calculation immediately/spontaneously take/perform an action
- For example, the vacuum agent whose agent function is tabulated is a simple reflex agent, because its decision is based only on the current location and on whether that location contains dirt.

• Example: simple reflex vacuum cleaner agent

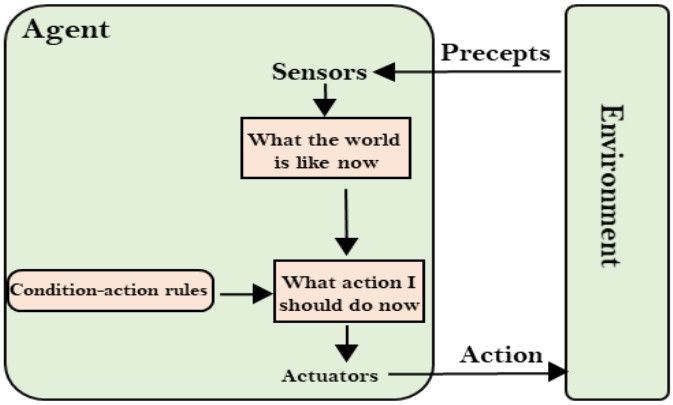
```
function REFLEX-VACUUM-AGENT([location,status]) returns an action
    if status = Dirty then return Suck
    else if location = A then return Right
    else if location = B then return Left
```

The Simple reflex agent works with agent function as Condition-action rule.

A condition-action rule is a rule that maps a state i.e, condition/current sate to an action. If the condition is true, then the action is taken, else not.

Such as a Room Cleaner agent, it works only if there is dirt in the room.

“The car in front is braking.” Then, this triggers some established connection in the agent program to the action “initiate braking.” We call such a connection a **condition–action rule**, written as **if** *car-in-front-is-braking* **then** *initiate-braking*.



Above gives the structure of this general program in schematic form, **Depicts how condition-action rule allow the Simple reflex agent to make the connection from percept to action**
The program above is specific to one particular vacuum environment. A more general and flexible approach is first to **build a general-purpose interpreter** for **condition– action** rules and **then to create rule sets** for specific task environments.

function SIMPLE-REFLEX-AGENT(*percept*) **returns** an action
persistent: *rules*, a set of condition–action rules

state ← INTERPRET-INPUT(*percept*)
rule ← RULE-MATCH(*state*, *rules*)
action ← *rule*.ACTION
return *action*

Figure 2.10 A simple reflex agent. It acts according to a rule whose condition matches the current state, as defined by the percept.

The INTERPRET-INPUT function generates an abstracted description of the current state from the percept, and the RULE-MATCH function returns the first rule in the set of rules that matches the given state description.
Put else part also for completeness

- Problems for the simple reflex agent design approach:
  - Simple-reflex agents are **simple,** but they turn out to be of **very limited intelligence.**
  - The agent will work only if the **correct decision** can be made on the basis of the **current percept** – that is only if the **environment is fully observable**.
  - If there occurs any change in the environment, then the collection of rules needs to be updated.
  - **Infinite loops** are often unavoidable when agents operating in partially observable environments – escape could be possible if the agent can randomize its actions.

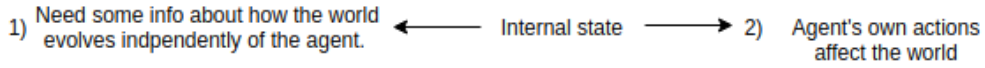
2. Model-based reflex agent – agent with memory

• For the world that is partially observable, the agent should keep track of the part of the world it can't see now

- That depends on the percept history
- Reflecting some of the unobserved aspects of current state

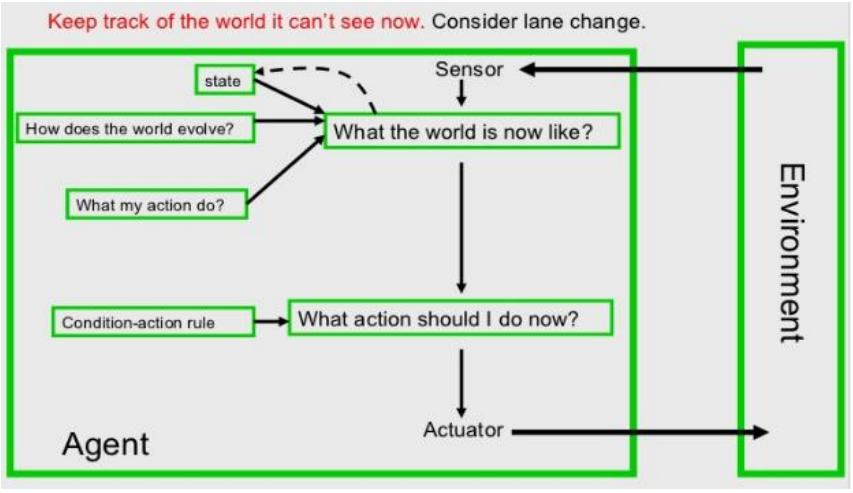
- E.g., driving a car and changing lane

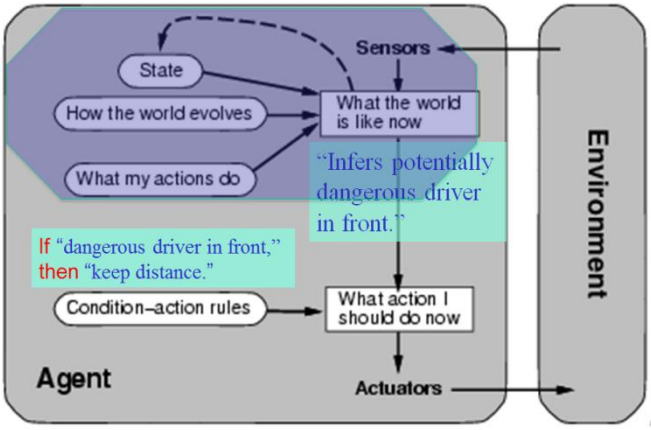
- The agent has to keep track of the internal state which is adjusted by each percept and that depends on the percept history. The current state is stored inside the agent which maintains some kind of structure describing the part of the world which cannot be seen.
- So, **Current State = Old Internal State + Current Percept**
- **Update Internal state** : Updating the internal state information as time goes by requires two kinds of knowledge to be encoded in the agent program
  – Information about how the world evolves independently of the agent
  – Information about how the agent's own actions affects the world



- A model-based agent has two important factors:
  - **Model:** it is the knowledge "about How the world works" Whether implemented in simple boolean circuits or in complete scientific theories is called the Model of the World. An agent that uses such a model is called a **Model-Based-Agent**.
  - **Internal State:** It is a representation of the current state based on percept history.
- These agents have the model, "which is knowledge of the world" and based on the model they perform actions.

Structure of the Model-based Reflex Agent





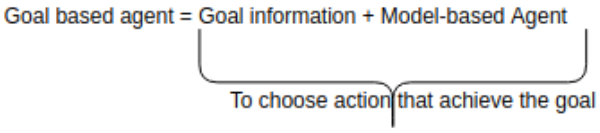
function MODEL-BASED-REFLEX-AGENT(*percept*) **returns** an action
persistent: *state*, the agent's current conception of the world state
model, a description of how the next state depends on current state and action
rules, a set of condition–action rules
action, the most recent action, initially none

state ← UPDATE-STATE(*state*, *action*, *percept*, *model*)
rule ← RULE-MATCH(*state*, *rules*)
action ← *rule*.ACTION
return *action*

Figure 2.12 A model-based reflex agent. It keeps track of the current state of the world, using an internal model. It then chooses an action in the same way as the reflex agent.

3. Goal-based agents

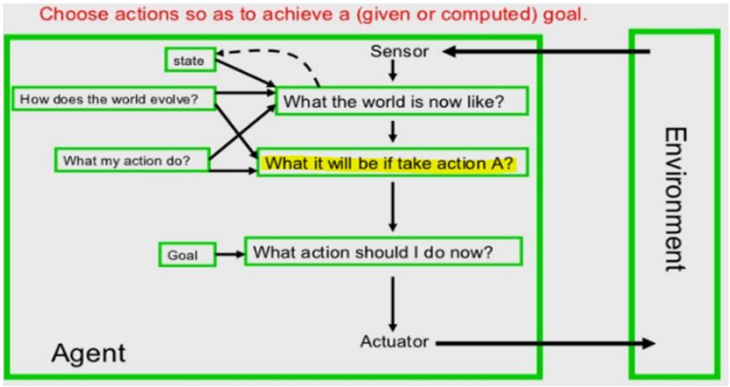
- In life, in order to get things done we set goals for us to achieve, this pushes us to make the right decisions when we need to.
- A simple example would be the shopping list; our goal is to pick up every thing on that list. This makes it easier to decide if you need to choose between milk and orange juice because you can only afford one. As milk is a goal on our shopping list and the orange juice is not we chose the milk.
  - **The knowledge of the current state environment is not always sufficient to decide for an agent to what to do.** (e.g. decision at a road junction)
  - Goal-based agents expand the capabilities of the model-based agent by having the "goal" information.



  - They choose an action, so that they can achieve the goal.
  - These kinds of agents take decisions based on how far they are currently from their **goal**(description of desirable situations). Their every action is intended to reduce its distance from the goal.
  - So in an intelligent agent having a set of goals with desirable situations are needed.
  - The agent program can use these goal information along with**set of possible actions** and their **predicted outcomes**in order to choosewhich action(s) achieve our goal(s).
  - The knowledge that supports its decisions is represented explicitly and can be modified, which makes these agents more flexible. They usually require

search and planning, which makes an agent proactive.. The goal-based agent's behavior can easily be changed.

- Unlike the previous reflex agents before just actingwhen condition matches, this agent reviews many/ a long sequence of possible actions and c**hooses the one which come closest to achieving its goals,**whereas the reflex agents just have an automated response for certain situations.



4. Utility-based agents

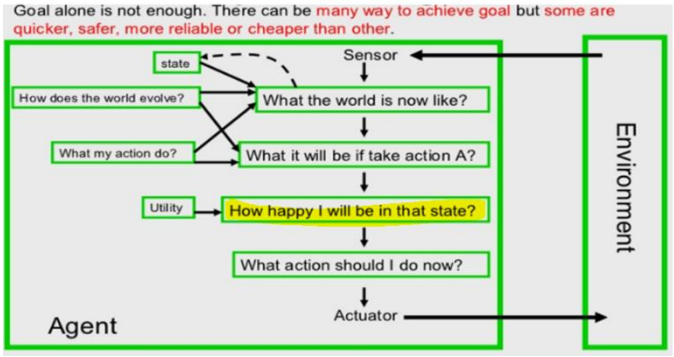
Sometimes achieving the desired goal is not enough. We may look for a **quicker, safer, cheaper trip** to reach a destination.

Agent happiness should be taken into consideration- they just provide a binary distinction between happy and unhappy states. E.g. meals in Canteen, good or not ?

Many action sequences -> the goals

- some are better and some worse
- If goal means success, then **utility** means the degree of success (how happy agent is if reached a goal state) =>driving car=>utility=> faster/ safer

- Utility describes how **"happy"** the agent is.
- Utility-based agent act based on not only goals but also the best way to achieve the goal.
- These agents are similar to the goal-based agent but provide an extra component of utility measurement which makes them different by providing a **measure of success at a given state.**
- When there are multiple possible alternatives, then to decide which action is best, utility-based agents are used. They choose actions based on a **preference (utility)** for each state
- The utility function maps each state to a real number which **describes the associateddegree of happiness.**



- A rational utility-based agent chooses the action that maximizes the expected utility of the actions outcomes.

5. Learning Agents

- After an agent is programmed, can it work immediately?
  - No, it still need teaching
- Turing – **instead of actually programming intelligent machines by hand, which is too much work, build learning machines and then teach them**
- In AI,
  - Once an agent is done
  - We teach it by giving it a set of examples
  - Test it by using another set of examples
- Based on performance of agent on unknown examples, We then say the agent learns
  - A learning agent

- A learning agent in AI is the type of agent which can learn from its past experiences, or it has learning capabilities.
- Learning agent starts to act with basic knowledge in initially unknown environments and then able to act and adapt automatically through learning and become more competent than its initial knowledge alone might allow.
- A learning agent has mainly four conceptual components, which are:

i. **Learning element:** It is responsible for making improvements by learning from environment

ii. **Performance element:** It is responsible for selecting an external action- it takes in percepts and decides on actions.

iii. **Critic:**The learning element takes feedback from critics which describes how well the agent is doing with respect to a fixed performance standardand determines how the performance element should be modified to do better in the future.
(Feedback from user or examples, good or not?)

iv. **Problem generator:** This component is responsible for suggesting actions that will lead to new and informative experiences.

Automated taxi example:

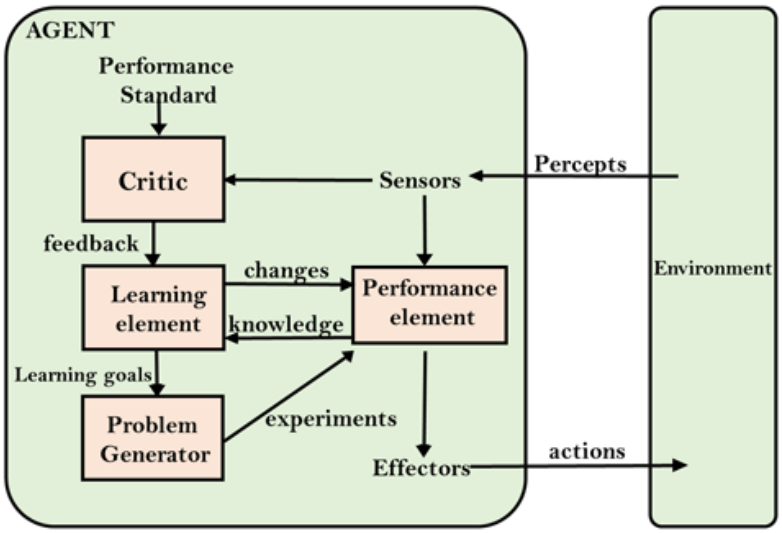
The performance element consists of whatever collection of knowledge and procedures the taxi has for **selecting its driving actions**. The taxi **goes out** on the road and drives, using this **performance element**. The critic **observes the world** and **passes information** along to the learning element.

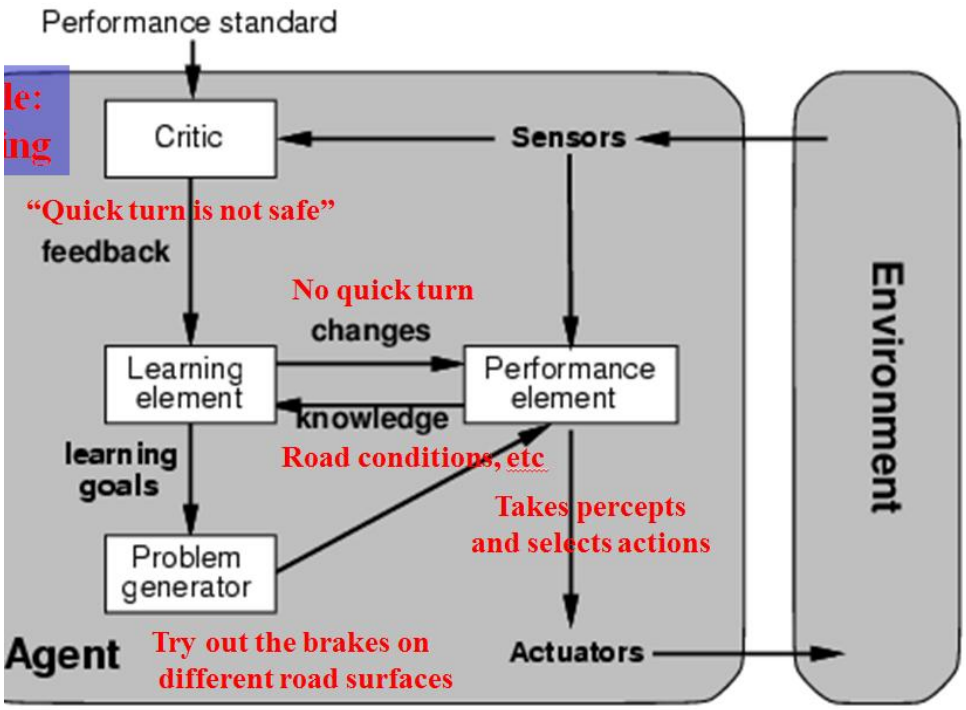
For example, after the taxi makes a quick left turn across three lanes of traffic, the **critic** observes the shocking language used by other drivers.

From this experience, the **learning element** is able to formulate a rule saying this was a bad action, and the **performance element is modified** by installation of the new rule.

The **problem generator** might identify certain areas of behavior in need of improvement and suggest experiments, such as trying out the brakes on different road surfaces under different conditions.

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





Steps performed by Problem-solving agent – formulate, search, execute

1. **Goal Formulation**
2. **Problem formulation**

   Components in problem definition

   a. **Initial State**
   b. **Actions**
   c. **Transition Model**
   d. **Goal Test**
   e. Path cost
3. **Search** for solution: Given the problem, search for a solution --- *a sequence of actions to achieve the goal starting from the initial state.=>explore the state space*
4. **Execution of the solution**

- **Goal Formulation:** It is the first and simplest step in problem-solving. It organizes the steps required to formulate one goal out of multiple goals and which requires some actions to achieve that goal. Goal formulation is based on the current situation and the agent's performance measure.
- **Problem formulation:** process of deciding what actions and states to consider, given a goal.

For now, let us assume that the agent will consider actions at the level of driving from one major town to another.

Each state therefore corresponds to being in a particular town.

Problem: On holiday in country Romania currently in Arad. Flight leaves tomorrow to Bucharest. Find a short route to drive to Bucharest.

Formulate problem:

states: various cities

actions: drive between cities

solution: sequence of cities, e.g., Arad, Sibiu, Fagaras, Bucharest

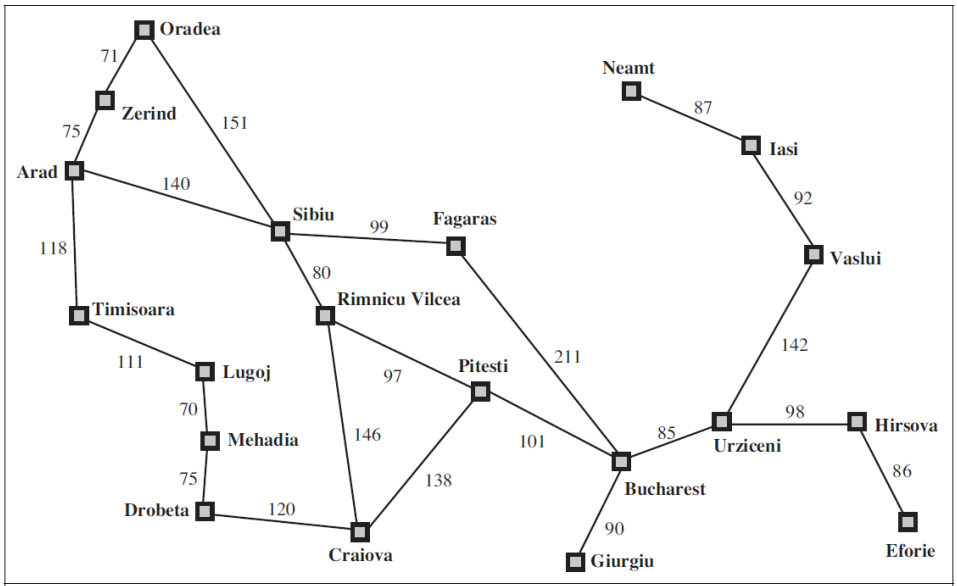


Figure 3.2 A simplified road map of part of Romania.

Properties of thebasic search Agent

Under the following assumptions about the environment, the solution to any problem is a fixed sequence of actions.

1. **Observable=>**agent always **knows the current state.**
   supposing that each city on the map has a sign indicating its presence to arriving drivers.
2. **Discrete =>only finite actions to choose from.**
   each city is connected to a small number of other cities.
3. **Known =>**agent knows **which states are reached by each action.**
4. **Deterministic =>each action has exactly one outcome.**

After **formulating a goal and a problem to solve**, the agent calls a **search** procedure to **solve it.** It then **uses the solution** to guide its **actions**, doing whatever the solution recommends as the next thing to do.

- **Search:** The process of **looking for a sequence of actions that reaches the goal i**s called search.
- **Solution:** A search algorithm takes a **problem as input**and **returns a solution**in the form of an **action sequence**. It finds the best algorithm out of various algorithms, which may be proven as the best optimal solution.
- **Execution:** Once a **solution is found, the actions**it recommends can be **carried out**called execution. It executes the best **optimal solution**found from the searching algorithms to reach the goal state from the current state.

Goal-based agents

- **Formulation**: goal and problem
- Given: initial state
- **Goal**: To reach the specified goal (a state) through the execution of appropriate actions.
- **Search** for a suitable action sequence and **execute the actions**

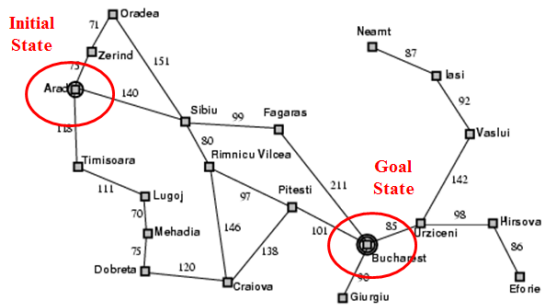
Formulate goal:
- **be in Bucharest (Romania)**

Formulate problem:
- **action: drive between pair of connected cities (direct road)**
- **state: be in a city (20 world states)**

Find solution:
- **sequence of cities leading from start to goal state, e.g., Arad, Sibiu, Fagaras, Bucharest**

Execution
- **drive from Arad to Bucharest according to the solution**

Example: Path Finding problem



Environment: fully observable (map), deterministic, and the agent knows effects of each action. Is this really the case?

Note: Map is somewhat of a "toy" example. Our real interest: *Exponentially large spaces*, with e.g. 10^100 or more states. Far beyond full search. Humans can often still handle those!
One of the mysteries of cognition.

Thus, we have a simple "formulate, search, execute" design for the agent, as shown in Figure 3.1.

```
function SIMPLE-PROBLEM-SOLVING-AGENT(percept) returns an action
  persistent: seq, an action sequence, initially empty
              state, some description of the current world state
              goal, a goal, initially null
              problem, a problem formulation

  state ← UPDATE-STATE(state, percept)
  if seq is empty then
      goal ← FORMULATE-GOAL(state)
      problem ← FORMULATE-PROBLEM(state, goal)
      seq ← SEARCH(problem)
      if seq = failure then return a null action
  action ← FIRST(seq)
  seq ← REST(seq)
  return action
```

Figure 3.1 A simple problem-solving agent. It first formulates a goal and a problem, searches for a sequence of actions that would solve the problem, and then executes the actions one at a time. When this is complete, it formulates another goal and starts over.

Well-defined problems and solutions

Components involved in problem formulation/definition

1. **Initial State:** It is the starting state or initial step of the agent towards its goal.
   For example, the initial state for our agent in Romania might be described as In(Arad).

2. **Actions/ Operators:** It is the description of the possible actions available to the agent. Given a particular state s,

   ACTIONS(s) returns the set of actions that can be executed in s. We say that each of these actions is **applicable** in s.

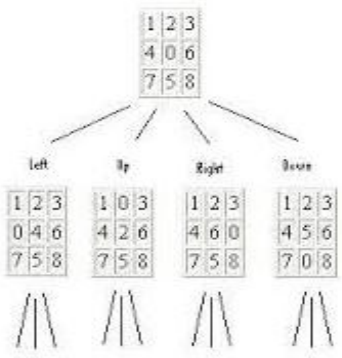
   For example, from the state In(Arad), the applicable actions are {Go(Sibiu), Go(Timisoara), Go(Zerind)}.

3. **Transition Model/ Successor function:** It describes what each action does.
   specified by a function RESULT(s, a) that returns the state that results from doing action ain state s. We also use the term **successor** to refer to any state reachable from a given state by a single action.

   For example, we have

RESULT(In(Arad),Go(Zerind)) = In(Zerind)

Together, the initial state, actions, and transition model implicitly define the **state space** of the problem—the set of all states reachable from the initial state by any sequence of actions. The state space **forms** a directed network or **graph** in which the nodes are states and the links between nodes are actions.



A **path** in the state space is a sequence of states connected by a sequence of actions.

4. **Goal Test:** It determines whether a given state is a goal state.Sometimes there is an explicit set of possible goal states, and the test simply checks whether the given state is one of them.
   The agent's goal in Romania is the singleton set {In(Bucharest)}.

Sometimes the goal is specified by an abstract property rather than an explicitly enumerated set of states. For example, in chess, the goal is to reach a state called "checkmate," where the opponent's king is under attack and can't escape.

Explicitenumerated set, e.g. In(Bucharest)
Implicitabstract, e.g. checkmate

5. **Path cost:** It assigns a numeric cost to each path to achieve the goal. The problem-solving agent selects a cost function, which reflects its own performance measure.
   S**tep cost** of taking action ain state sto reach state s'is denoted by
   c(s, a,s'). We assume that step costs are non-negative.

   A**n optimal solution has the lowest path cost among all the solutions.**

- The preceding elements (initial state, actions, transition model, goal test, path cost) define a **problem** and can be gathered into a single data structure that is given as **input to a problem-solving algorithm**.

- A **solution**to a problem is an **action sequence** that leads from the initial state to a goal state. Solution quality is measured by the path cost function, and an **optimal solution** has the lowest path cost among all solutions.

Formulating Problems

Road map

a formulation of the problem of getting to Bucharest in terms of the initial state, actions, transition model, goal test, and path cost. This formulation seems reasonable, but it is still a *model*—an abstract mathematical description.

- Real world is complex and has more details

- Irrelevant details should be removed from state space and actions, which is called abstraction.
- What's the appropriate level of abstraction?
    - the abstraction is valid, if we can expand it into a solution in the **more detailed** world
    - the abstraction is useful, if carrying out the actions in the solution is **easier than the original problem.**

State space representation

- A **state–space representation** allows for the formal definition of a problem, which makes the movement from initial state to the goal state quite easily. So, we can say that various problems like planning, learning, theorem proving, etc., are all essentially search problems only.
- **State–space search** is a process used in the field of computer science, including AI, in which successive configurations or states of an instance are considered, with the goal of finding a goal state with a desired property.

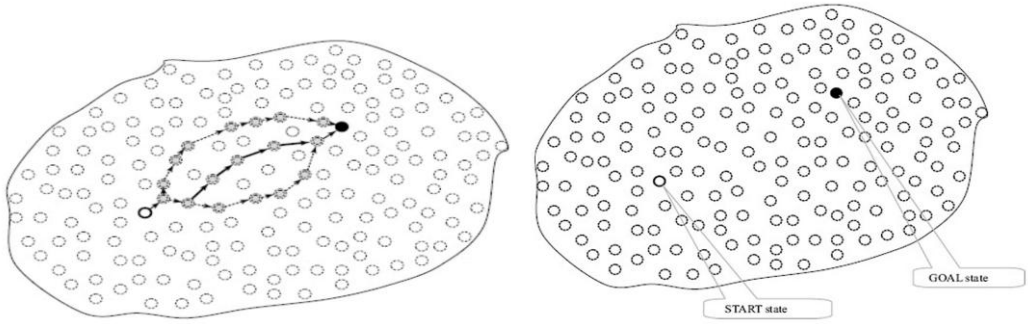


Fig: The state space

FIGURE 2.4 The solution is a sequence of moves that end in the GOAL state. There may be more than one solution to a problem. The figure shows one solution with thick arrows, and two alternatives with dotted arrows.

Example problems

Basically, there are two types of problem approaches:

- ***Toy problems***
    - those intended to illustrate or exercise various problem-solving methods
    - E.g., puzzle, chess, etc.
- ***Real-world problems***
    - tend to be more difficult and whose solutions people actually care about
    - E.g., Design, planning, etc.

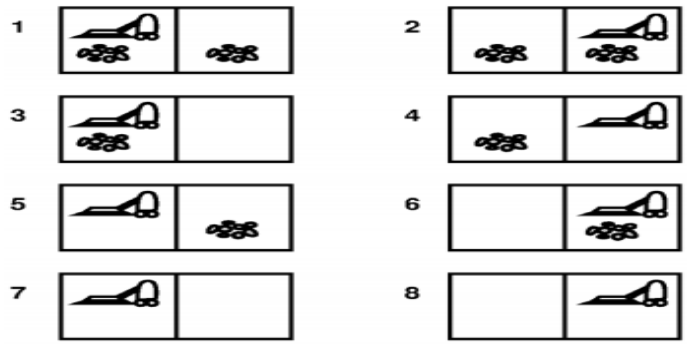
Toy Problems

Vacuum World

States: The state is determined by both the agent **location** and the **dirt** locations.

The agent is in one of two locations A or B, hence 2 possibilities.

Dirt can be in none, one(A/B)or both location, and hence 4 possibilities. Thus, there are total $2 \times 2^2 = 8$ possible world states. For environment with n locations has $(n * 2^n)$ states.



• **Initial state**: Any state can be designated as the initial state.

• **Actions**: In this simple environment, each state has just three actions: move *Left*, *Right*, and *Suck/clean*. Larger environments might also include *Up* and *Down*.

• **Transition model**:

- This is a function which define transition from one state to another. Consider state 1 in following diagram is initial state. There 3 actions possible
  - If it move LEFT (L) then it will remain in same state
  - and If it move RIGHT (R), agent will be in state 2
  - If it suck or clean, it will be in state 3

In this way complete graph is created. Transition function be defined as

state=function(*state*,*action*)➔3=*function*(1,*S*)

The complete state space is shown in Figure

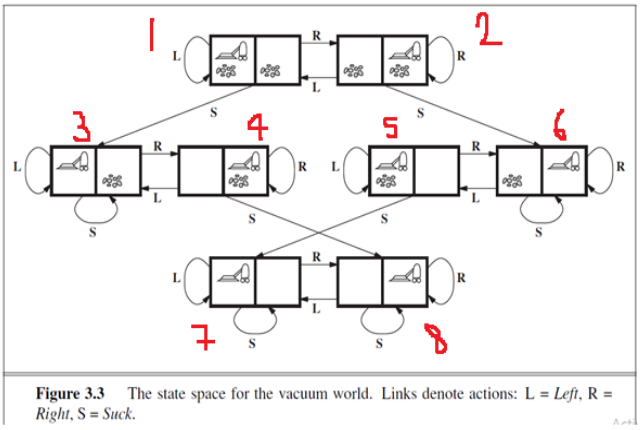


Figure 3.3 The state space for the vacuum world. Links denote actions: L = *Left*, R = *Right*, S = *Suck*.

• **Goal test**: This checks whether all the squares are clean.

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

Gaming=>8 Puzzle or Slide Puzzle

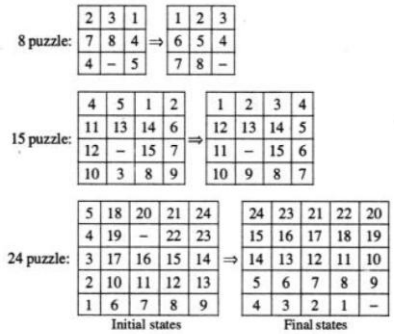
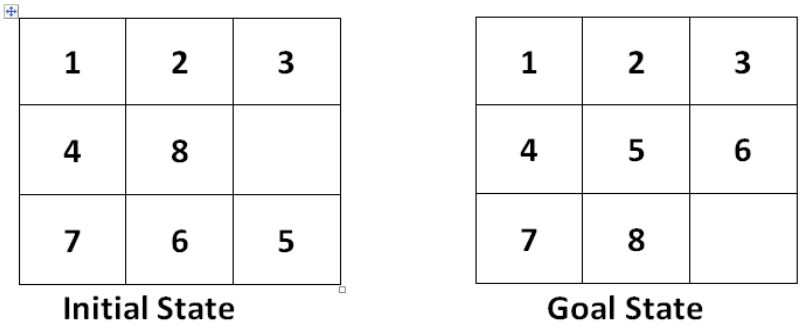


Figure 3.2 State space representation of the 8, 15 and 24 puzzle problems.

The 8-puzzle, an instance of which is shown in Figure

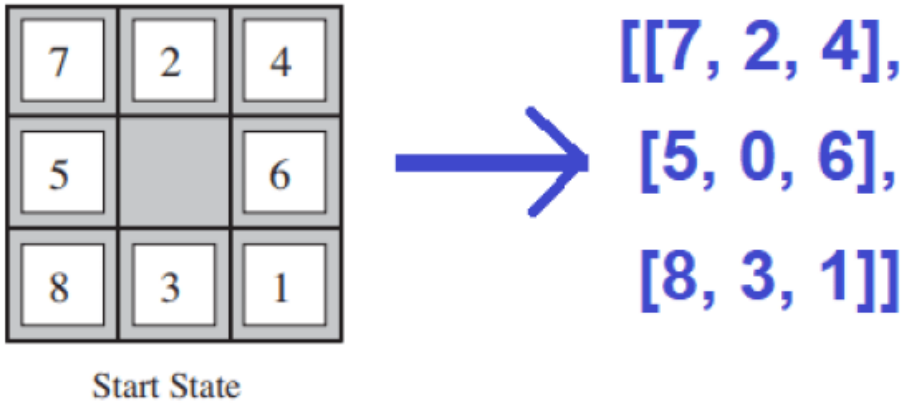


Consists of a 3×3 board with eight numbered tiles and a blank space.

- A tile adjacent to the blank space can slide into the space.
- The objective is to reach a specified goal state, such as the one shown on the right of the figure.
- Our task is to convert the current (Start) state into goal state by sliding digits into the blank space. The standard formulation is as follows.

State Representation

Any state can be represented by 2 dimensional array where each value is the name of tile and 0 shows empty tile. Simple Python representation of given initial state can be:-



States: A state description specifies the location of each of the eight tiles and the blank in one of the nine squares.

• **Initial state**: Any state can be designated as the initial state.

• **Actions**: The simplest formulation defines the actions as movements of the blank space slide *Left*, slide *Right*, slide *Up*, or slide *Down*. No diagonal movements assumption.

Possible Actions

It is define by empty tile position, possible actions are 2, 3 and 4, how?

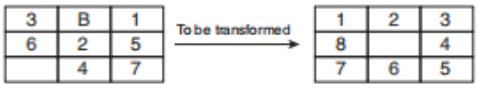
• **Transition model**: Given a state and action, this returns the resulting state;

Initial	{(1,2,3), (4,8,0), (7,6,5)}
Down	{(1,2,3), (4,8,5), (7,6,0)}
Left	{(1,2,3), (4,8,5), (7,0,6)}
Up	{(1,2,3), (4,0,5), (7,8,6)}
Right	{(1,2,3), (4,5,0), (7,8,6)}
Down	{(1,2,3), (4,5,6), (7,8,0)}

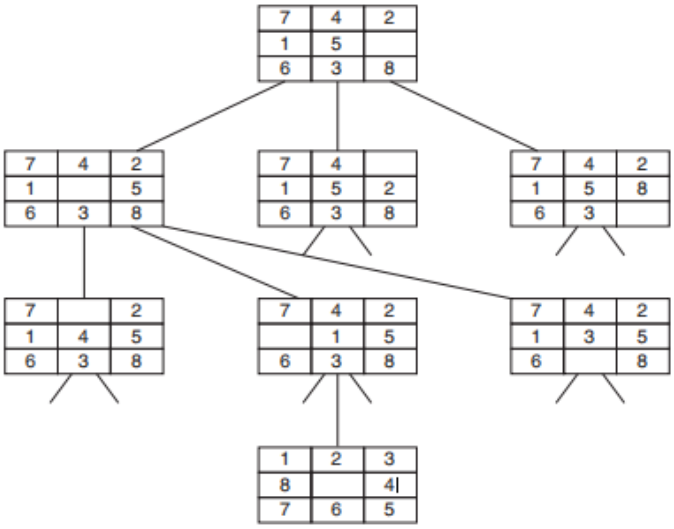
- **Goal test**: This checks whether the state matches the goal configuration shown in Figure above (Many other goal configurations are possible.)

• **Path cost**: Each step costs 1, so the path cost is the number of steps in the path. For our example path cost =5.

Note: The 8-puzzle problem is a type of **sliding-block problem** which is used for test problems for new search algorithms in artificialintelligence.

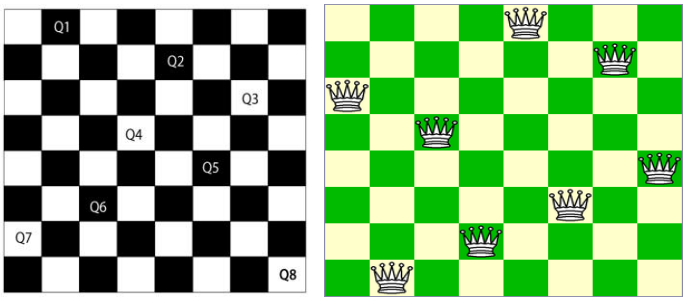


The states of eight-tile puzzle are the different permutations of the tiles within frame.
Let's do a standard formulation of this problem now.



Logical Puzzle Solving

- The goal of the **8-queens problem** is to place eight queens on a chessboard such that **no queen attacks any other**. (A queen attacks any piece in the same row, column or diagonal.)
- From the following figure, we can understand the problem as well as its one of the correct solution.



For this problem, there are two main kinds of formulation:

- **Incremental formulation:** It **starts from an empty state**where the operator adds a queen onto board at each step.

Following steps are involved in thisformulation:

- **States**: Any arrangement of 0 to 8 queens on the board is a state.
- **Initial state**: No queens on the board.
- **Actions**: Add a queen to any empty square.
- **Transition model**: Returns the board with a queen added to the specified square.
- **Goal test**: 8 queens are on the board, none attacked.

Path cost: There is no need for path cost because only final states are counted.

In this formulation, we have **64**(no queen placed) · **63**(already a queen placed) · · · **57**(q1,…q7 placed=64-7=57)≈ $\mathbf{1.8\times10^{14}}$possible sequences to investigate.

- **Complete-state formulation:** It **starts with all the 8-queens** on the chessboard and moves them around, saving from the attacks.
  - starts with all 8 queens on the board
  - move the queens individually around
  - States:
    - any arrangement of 8 queens, one per column in the leftmost columns
  - Operators: move an attacked queen to a row, not attacked by any other

Following steps are involved in this formulation

A better formulation would prohibit placing a queen in any square that is **already attacked**

- **States**: All possible arrangements of n queens ($0 \leq n \leq 8$), one per column in the leftmost n columns, with no queen attacking another.
- **Actions**: Add a queen to any square in the leftmost empty column such that it is not attacked by any other queen.

This formulation reduces the 8-queens state space from $1.8\times10^{14}$to just 2,057, and solutionsare easy to find. On the other hand, for 100 queens the reduction is from roughly 10^{400} states to about 10^{52}states.

Problem formulation of the Water Jug Problem

Problem definition: In the **water jug problem in Artificial Intelligence**, we are provided with two jugs: one having the capacity to hold 3 gallons of water and the other has the capacity to hold 4 gallons of water. There is no other measuring equipment available and the jugs also do not have any kind of marking on them. So, the agent's task here is to fill the 4-gallon jug with 2 gallons of water by using only these two jugs and no other material. Initially, both our jugs are empty.

Problem formulation Steps:

1. **Describe the states**
2. **Identify the initial state**
3. **Identify the goal state**
4. **Identify set of rules (all possible actions)**
5. **Find the solution path in the state space**

- **Solution: Now, we need to solve the problem such that there is two gallons of water in four gallons of jug.** We will have to keep track of the amount of water in each of the jugs after we perform a particular action. Hence, we should represent the state such that it reflects the amount of water in each of the individual jugs.
- **Representation of state: For the aforementioned problem, the state can be described as ordered pair of** integer (*x, y) such that 'x' is the amount of water in four gallon of jug, hence 'x' can take values 0, 1, 2, 3, 4* gallons. (**Note: Since *'x' is the amount of water in four gallon of jug, the jug can be empty, i.e. has 0 gallons***of water or can have 1/2/3/4 gallons of water, respectively). '*y' is the amount of water in three gallons of jug.* Similarly *y can take values as 0, 1, 2, 3 ...*
- **Initial state: Initially, we assume that both the jugs are empty. That is both the jugs contain zero gallons of** water. This initial situation (also called as initial state) can be represented using state representation notation as follows:
- SI = (0, 0) Note that initial state is represented by SI.
- **Goal state: We need two gallons of water in four gallons of jug as the goal but there can be any amount of** water in three gallons of jug. Hence goal state is the set of states.
- SG = {(2, 0), (2, 1), (2, 2), (2, 3)}
- This notation indicates that there must be two gallons of water in four gallons of jug but there can be 0/1/2/3 gallons of water in three gallons of jug which we are not concerned. Hence, goal state represented by SG is set of states as shown. Hence, as the goal the agent can achieve any one of the states that belongs to this set.
- **Set of rules: Now, we write set of rules that will be applied to current state so as to change the current state** and go to the next state. To write these set of rules one has to simply think of **all possible combinations** that can occur,

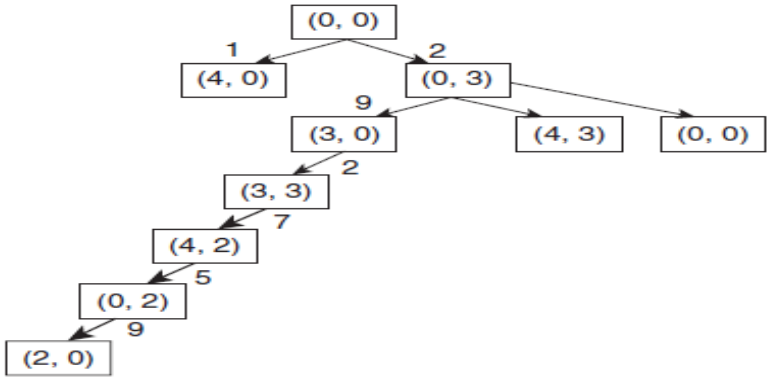
Sr.	Current State	Next State	Descriptions
1	(x, y) if x < 4	(4, y)	Fill the 4 gallon jug
2	(x, y) if y < 3	(x, 3)	Fill the 3 gallon jug
3	(x, y) if x > 0	(x - d, y)	Pour some water out of the 4 gallon jug
4	(x, y) if y > 0	(x, y - d)	Pour some water out of the 3 gallon jug
5	(x, y) if y > 0	(0, y)	Empty the 4 gallon jug
6	(x, y) if y > 0	(x, 0)	Empty the 3 gallon jug on the ground
7	(x, y) if x + y >= 4 and y > 0	(4, y - (4 - x))	Pour water from the 3 gallon jug into the 4 gallon jug until the 4 gallon jug is full
8	(x, y) if x + y >= 3 and x > 0	(x - (3 - y), 3)	Pour water from the 4 gallon jug into the 3-gallon jug until the 3 gallon jug is full
9	(x, y) if x + y <= 4 and y > 0	(x + y, 0)	Pour all the water from the 3 gallon jug into the 4 gallon jug
10	(x, y) if x + y <=3 and x > 0	(0, x + y)	Pour all the water from the 4 gallon jug into the 3 gallon jug
11	(0, 2)	(2, 0)	Pour the 2 gallons from 3 gallon jug into the 4 gallon jug
12	(2, y)	(0, y)	Empty the 2 gallons in the 4 gallon jug on the ground

The listed production rules contain all the actions that could be performed by the agent in transferring the contents of jugs. But, to solve the water jug problem in a minimum number of moves, following set of rules in the given sequence should be performed:

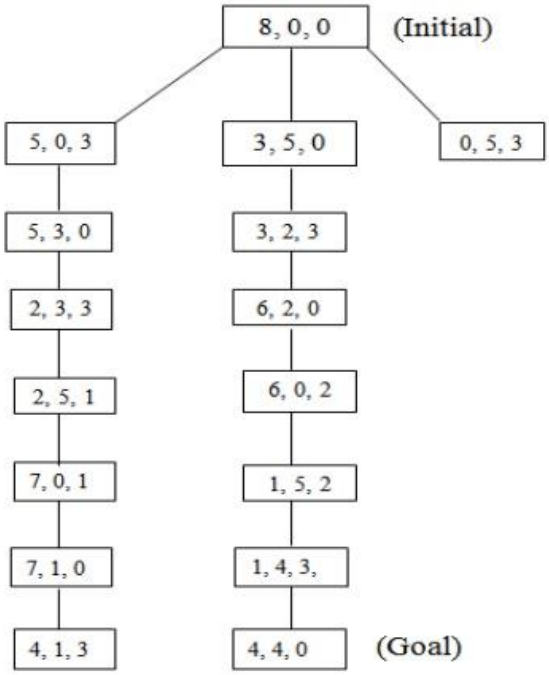
Solution of water jug problem according to the production rules:

4 gallon jug contents	**3 gallon jug contents**	**Rule followed**
0 gallon	0 gallon	Initial state
0 gallon	3 gallons	Rule no.2
3 gallons	0 gallon	Rule no. 9
3 gallons	3 gallons	Rule no. 2
4 gallons	2 gallons	Rule no. 7
0 gallon	2 gallons	Rule no. 5
2 gallons	0 gallon	Rule no. 9

we will apply these set of rules to the initial state to generate the goal state. These set of rules will be applied to each state until we reach the goal. Hence, we generate a tree which represents all the possible states in the problem, and this is called as *state space. In a state space, we will find a path* to the goal this is called a *solution path. Solution path is found from state space by state space search.* This approach of solution is called as *state-space approach.* **Solution Tree: To represent all the possible states in the state space.**



Other with 3 gallons 5L, 3L, 8L



Figure

Real-world problem

- **Route finding problems**

Route-finding algorithms are used in a variety of applications. Some, such as **Web sites** and **in-car systems** that **provide driving directions,** are relatively straightforward extensions of the Romania example. Others, such as r**outing** video streams in computer networks, military operations **planning**, and airline **travel-planning** systems, involve much more complex specifications. Consider the airline travel problems that must be solved by a travel-planning Web site:

• **States**: Each state obviously includes a location (e.g., an airport) and the current time. Furthermore, because the cost of an action (a flight segment) may depend on previoussegments, their fare bases, and their status as domestic or international, the state must record extra information about these "historical" aspects.

• **Initial state**: This is specified by the user's query.

• **Actions**: Take any flight from the current location, in any seat class, leaving after thecurrent time, leaving enough time for within-airport transfer if needed.

• **Transition model**: The state resulting from taking a flight will have the flight's destinationas the current location and the flight's arrival time as the current time.
• **Goal test**: Are we at the final destination specified by the user?
• **Path cost**: This depends on monetary cost, waiting time, flight time, customs and immigrationprocedures, seat quality, time of day, type of airplane, frequent-flyer mileageawards, and so on.

- **Touring problems** are closely related to route-finding problems, but with an important difference. Consider, for example, the problem "**Visit every city** in Romania **at least once**, **starting and ending in Bucharest**." As with route finding, the actions correspond to **trips between adjacent cities.** The state space, however, is quite different. Each state must **include** not just the **current location**but also the *set of cities the* **agent has visited**.
- So the initial state would be **In(Bucharest), Visited({Bucharest**}), a typical intermediate state would be In(Vaslui), Visited({Bucharest , Urziceni, Vaslui}), and thegoal test would check whether the agent is in Bucharest and all 20 cities have been visited.

- **Travelling salesman problem-**
  The **traveling salesperson problem** (TSP) is a touring problem in which each city must be **visited exactly once.** The aim is to **find the *shortest* tour.** In addition to planning trips for traveling salespersons, these algorithms have been used for tasks such as planning movements of automatic circuit-board drills and of stocking machines on shop floors.
- **VLSI Layout problem:** related to the positioning of various components and connections on the chipin order to minimize the area, circuit-delays, stray-capacitances, and maximizing the manufacturing yield.

The layout problem is split into two parts:

- **Cell layout:** Here, the primitive components of the circuit are grouped into cells, each performing its specific function. Each cell has a fixed shape and size. The task is to place the cells on the chip without overlapping each other.
- **Channel routing:** It finds a specific route for each wire through the gaps between the cells.
- **Robot navigation** is a generalization of the route-finding problem. Rather than following a discrete set of routes, a robot **can move in a continuous space** with (in principle) an **infinite set of possible actions and states**.
  - For a **circular robot** moving on a **flat surface**, the space is essentially **two-dimensional.**
  - When the robot has **arms and legs or wheels** that must also be **controlled**, the search space becomes **many-dimensional.**
  - Advanced techniques are required just to make the **search space finite.**

Searching for solutions=>**SEARCH TREE AND SEARCH GRAPH**

- **Finding out a solution is done by**
  - **searching through the state space**
  - **All problems are transformed**
    - **as a search tree**
    - **generated by the initial state and successor function**

Having formulated some problems, we now need to solve them.

A **solution**is an action sequence, so search algorithms work by considering various possible action sequences.

The possible action sequences starting at the initial state form a search treewith the

- **initial state** at the root;
- the branches are **actions** and
- the**nodes**correspond to **states in the state space** of the problem.

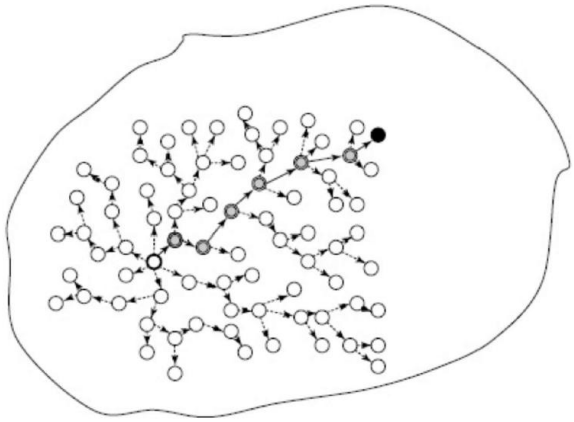
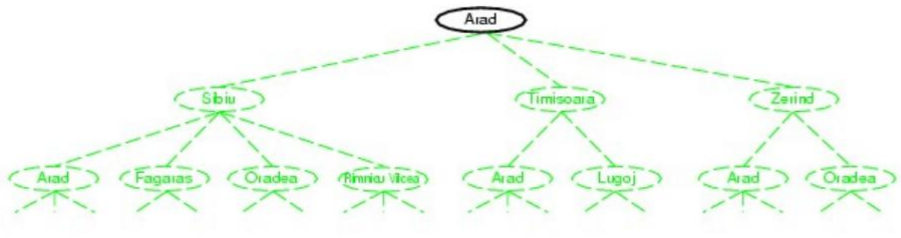


FIGURE 2.7 The nodes visited by a search algorithm, form a search tree shown by empty circles and dotted arrows. The solution found is shown with shaded nodes and solid arrows.

Figure 3.6 shows the first few steps in growing the search tree for finding a routefrom Arad to Bucharest.

Tree search example

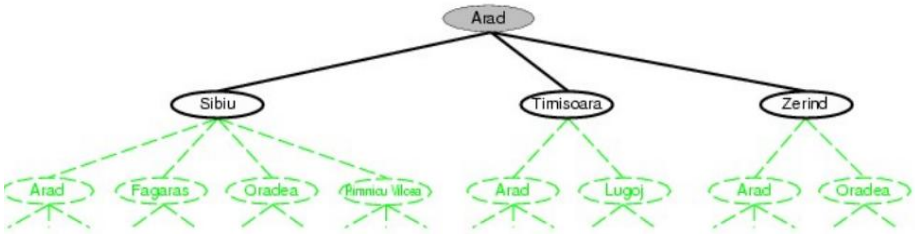
- The root node of the tree corresponds to the initial state, *In(Arad).*



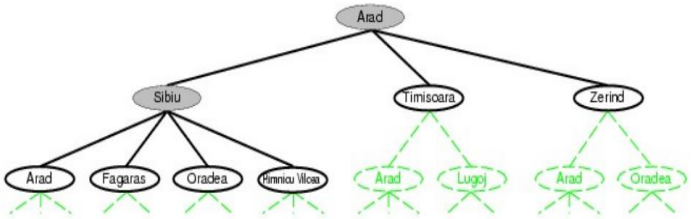
The first step is to test whether this is a **goal** state.

Then weneed to consider taking various **actions.** We do this by **expanding**the current state; that is,applying each legal action to the current state, thereby **generating**a new set of states.

In this case, we add three branches from the **parent node** *In(Arad)* leading to three new **childnodes**: *In(Sibiu), In(Timisoara),* and *In(Zerind).*



Now we must choose which of these three possibilities to consider further.
This is the essence of search—following up one option now and putting the others asidefor later, in case the first choice does not lead to a solution. Suppose we choose Sibiu first.



Note: Arad added (again) to tree!(reachable from Sibiu)
Not necessarily a problem, butin Graph-Search, we will avoidthis by maintaining an"explored" list.

We check to see whether it is a goal state (it is not) and then expand it to get *In(Arad)*, *In(Fagaras)*, *In(Oradea)*, and *In(RimnicuVilcea)*.
We can then choose any of these four or go back and choose Timisoara or Zerind.
Each of these six nodes is a **leaf node**, that is, a node with no children in the tree.

- The set of all leaf nodes available for expansion at any given point is called the **frontier/open list/fringe**.

In Figures above, the frontier of each tree consists of those nodes with bold outlines.

- The process of expanding nodes on the frontier continues until **either a solution is found** or there are **no more states to expand**.

Search tree summary

- **Initial state**
  - **The root of the search tree is a search node**
- **Expanding**
  - **applying successor function to the current state**
  - **thereby generating a new set of states**
- **leaf nodes**
  - **the states having no successors**

  Fringe :Set of search nodes that have not been expanded yet.

The general TREE-SEARCH algorithm is shown informally in Figure 3.7.

function **TREE-SEARCH**(*problem, strategy*) return **a solution or failure**
  Initialize frontier to the *initial state* of the *problem*
  do
    if **the frontier is empty** then return ***failure***
    choose leaf node for expansion according to strategy & remove from frontier
    if **node contains goal state** then return ***solution***
    else **expand the node and add resulting nodes to the frontier**

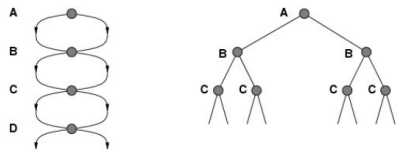
Determines search process!!

Search algorithms all share this basic structure; they vary primarily according to **how they choose which state to expand next**—the so-called **search strategy.**

one peculiar thing about the search tree shown in Figure expanded tree: it includes the path from Arad to Sibiu and back to Arad again! We say that *In(Arad)* is a **repeated state** in the search tree, generated in this case by a **loopy path**.
Considering such loopy paths means that the **complete search tree for Romania is *infinite***because there is no limit to how often one can traverse a loop even though the search space is limited.

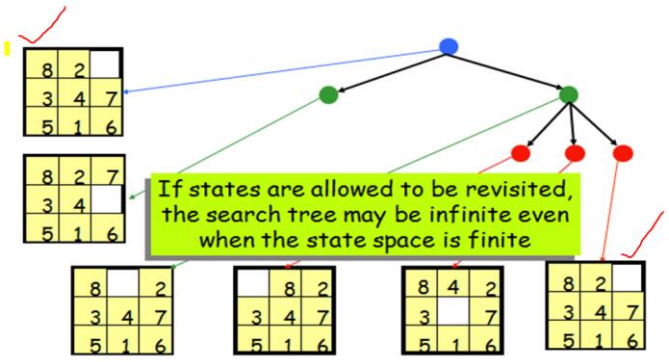
Avoiding Repeated States

- Failure to detect repeated states can turn a linear problem into an exponential one!
- Algorithms that forget their history are doomed to repeat it



state space size: d + 1 → search tree leaves: 2^d

Loopy paths are a special case of the more general concept of **redundant paths**, whichexist whenever there is more than one way to get from one state to another (for example, Arad — Sibiu and Arad — Zerind — Oradea — Sibiu). Need to reformulate the problem then each state can be reached only through one path.



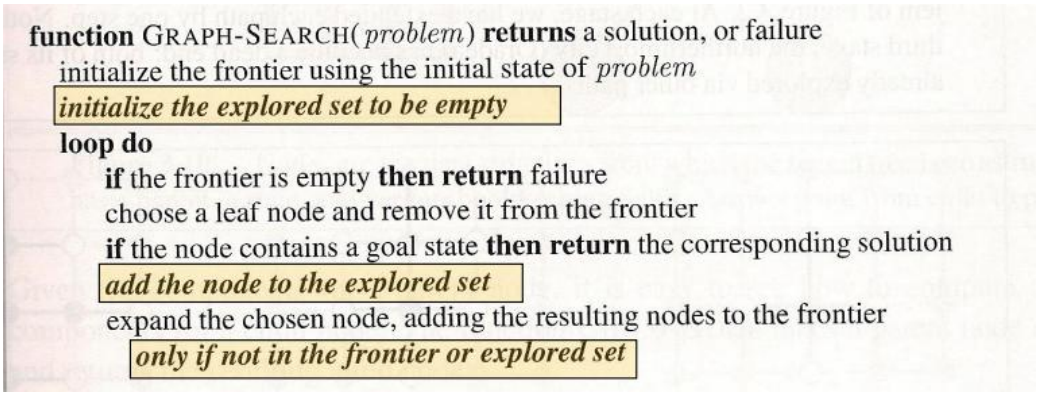
In other cases, redundant paths are unavoidable. This includes all problems where the actions are reversible, such as route-finding problems and sliding-block puzzles.

The redundant path situation occurs in almost every problem, and often makes the solution algorithm less efficient, worsening the performance of the searching agent.

- One way to **eliminate the redundancy** is to utilize the advantage given by the problem definition itself. For example, in the case of traveling from Arad to Bucharest, since the path costs are additive and step costs are non-negative, **only one path**among the various redundant paths **has the least cost**(and it is the shortest distance between the two states), and loopy paths are never better than the same path with loops removed.
- Another idea to avoid exploring redundant paths is to remember which states have been visited previously. Along with the search tree, an **explored set** is maintained which contains all the states previously visited. Newly generates which matches the previously generated nodes can be discarded.
    - In this way, every step moves the states in the frontier into the explored region, and some states in the unexplored region into the frontier, until the solution is found.

To do this, weaugment the TREE-SEARCH algorithm with a data structure called the **explored set/closed list**which remembers every expanded node.

- Newly generated nodes that **match previously generated nodes**—ones in the explored set or the frontier—can be **discarded** instead of being **added to the frontier.**
- The new algorithm, called GRAPH-SEARCH, is shown informally in Figure



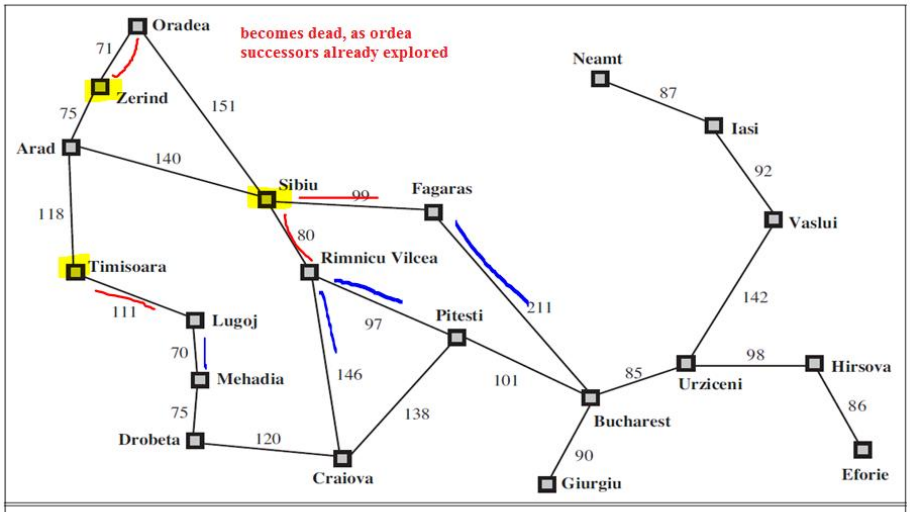


Figure 3.2 A simplified road map of part of Romania.

Clearly, the search tree constructed by the GRAPH-SEARCH algorithm contains at mostone copy of each state, so we can think of it as growing a tree directly on the state-space graph,as shown in Figure

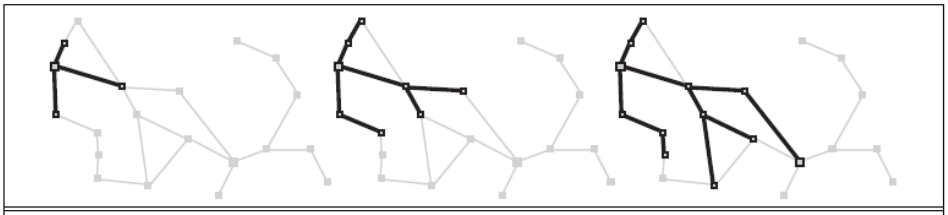
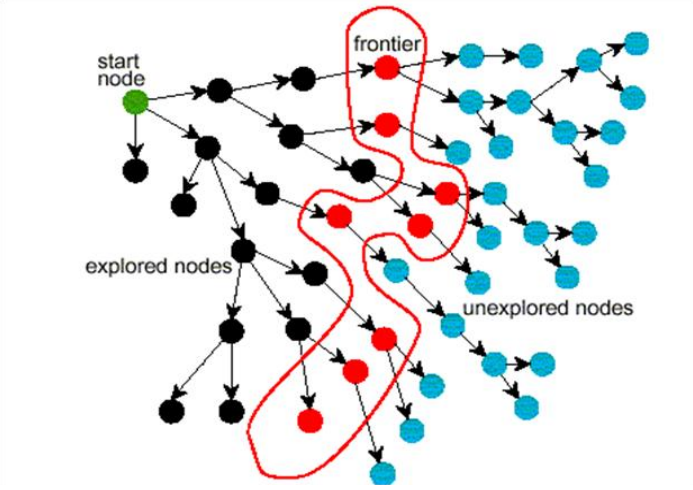


Figure 3.8 A sequence of search trees generated by a graph search on the Romania problem of Figure 3.2. At each stage, we have extended each path by one step. Notice that at the third stage, the northernmost city (Oradea) has become a dead end: both of its successors are already explored via other paths.

The algorithm has another nice property: the frontier **separates** the state-space graph into the explored region and the unexplored region, so that every path from the initial state to an unexplored state has to pass through a state in the frontier.



As every step moves a state from the frontier into the explored region while moving somestates from the unexplored region into the frontier, we see that the algorithm is *systematically*examining the states in the state space, one by one, until it finds a solution.

We can evaluate an algorithm's performance in four ways:

Completeness: Is the algorithm **guaranteed to find a solution** when there is one?
Optimality: Does the strategy **find the optimal solution**.
Time complexity: How long does it take to **find a solution**?
Space complexity: How much **memory** is needed to **perform the search**?

- In AI, the graph is often represented *implicitly* by the initial state, actions, and transition model and is frequently infinite. For these reasons, **complexity is expressed in terms of three quantities**:
  - **b, the branching factor** or maximum number of successors of any node;
  - **d, the depth** of the shallowest goalnode (i.e., the number of steps along the path from the root); and
  - **m, the maximum length**of any path in the state space.

- Time is often measured in terms of the **number of nodes generated** during the search, and space in terms of the maximum number of nodes **stored in memory.**

To assess the effectiveness of a search algorithm, **total cost**, which combines the search cost (depends on the time complexity) and the path cost(memory usage)of the solution found.
Ex:
For the problem of finding a route from Arad to Bucharest,

- the search cost is the **amount of time taken by the search** and
- the solution cost is the **total length of the path in kilometers**.
- Thus, to compute the total cost, we have to add milliseconds and kilometers.