

Artificial Intelligence and Machine Learning

UNIT-I:

- Introduction: Artificial Intelligence, AI Problems, AI Techniques, the Level of the Model, Criteria for Success. Defining the Problem as a State Space Search, Problem Characteristics, Production Systems, Search: Issues in the Design of Search Programs, Un-Informed Search, BFS, and DFS.
- Heuristic Search Techniques: Generate- And- Test, Hill Climbing, Best-First Search, A* Algorithm, Problem Reduction, AO* Algorithm

Introduction

- Since the invention of computers or machines, their capability to perform various tasks went on growing exponentially.
- Humans have developed the power of computer systems in terms of their diverse working domains, their increasing speed, and reducing size with respect to time.
- A branch of Computer Science named *Artificial Intelligence* pursues creating the computers or machines as intelligent as human beings.

What is AI?

- According to the father of Artificial Intelligence, John McCarthy, it is “*The science and engineering of making intelligent machines, especially intelligent computer programs*”.
- Artificial Intelligence is a way of **making a computer, a computer-controlled robot, or a software think intelligently**, in the similar manner the intelligent humans think.

AI is accomplished by studying how human brain thinks, and how humans learn, decide, and work while trying to solve a problem, and then using the outcomes of this study as a basis of developing intelligent software and systems

- **Machine Learning(ML)** : ML is an application of AI that provides systems the ability to automatically learn and improve from experience without being explicitly programmed.
- ML focuses on the development of computer programs that can access data and use it learn for themselves.
- **Deep Learning(DL)** : DL is a part of broader family of ML methods based on learning data representations as opposed to task specific algorithms.

- Learning can be supervised, semi-supervised and unsupervised.
- **Natural language Processing(NLP):** NLP is the ability of a computer program to understand human language as it is spoken. NLP is a component of AI.
- **Expert System** is a computer system that emulates the decision making ability of human expert.

Philosophy of AI

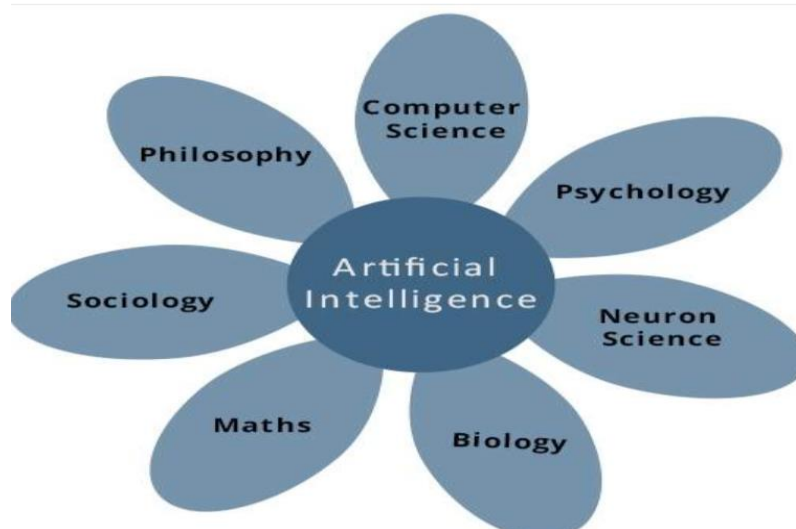
- While exploiting the power of the computer systems, the curiosity of human, lead him to wonder, *“Can a machine think and behave like humans do?”*
- Thus, the development of AI started with the intention of creating similar intelligence in machines that we find and regard high in humans.

Goals of AI

- **To Create Expert Systems** – The systems which exhibit intelligent behavior, learn, demonstrate, explain, and advice its users.
- **To Implement Human Intelligence in Machines** – Creating systems that understand, think, learn, and behave like humans.

What Contributes to AI?

- Artificial intelligence is a science and technology based on disciplines such as Computer Science, Biology, Psychology, Linguistics, Mathematics, and Engineering. A major thrust of AI is in the development of computer functions associated with human intelligence, such as reasoning, learning, and problem solving.
- Out of the following areas, one or multiple areas can contribute to build an intelligent system.



Programming Without and With AI

Programming Without AI	Programming With AI
A computer program without AI can answer the specific questions it is meant to solve.	A computer program with AI can answer the generic questions it is meant to solve.
Modification in the program leads to change in its structure.	AI programs can absorb new modifications by putting highly independent pieces of information together. Hence you can modify even a minute piece of information of program without affecting its structure.
Modification is not quick and easy. It may lead to affecting the program adversely.	Quick and Easy program modification.

What is AI Technique?

- In the real world, the knowledge has some unwelcomed properties –
 - Its volume is huge, next to unimaginable.
 - It is not well-organized or well-formatted.
 - It keeps changing constantly.
- AI Technique is a manner to organize and use the knowledge efficiently in such a way that –
 - It should be perceivable by the people who provide it.
 - It should be easily modifiable to correct errors.
 - **It should be useful in many situations though it is incomplete or inaccurate.**

Applications of AI

AI has been dominant in various fields such as –

- **Gaming** – AI plays crucial role in strategic games such as chess, poker, tic-tac-toe, etc., where machine can think of large number of possible positions based on heuristic knowledge.

- **Natural Language Processing** – It is possible to interact with the computer that understands natural language spoken by humans.
- **Expert Systems** – There are some applications which integrate machine, software, and special information to impart reasoning and advising. They provide explanation and advice to the users.
- **Vision Systems** – These systems understand, interpret, and comprehend visual input on the computer. For example,
 - Police use computer software that can recognize the face of criminal with the stored portrait made by forensic artist.
- **Speech Recognition** – Some intelligent systems are capable of hearing and comprehending the language in terms of sentences and their meanings while a human talks to it. It can handle different accents, slang words, noise in the background, change in human's noise due to cold, etc.
- **Handwriting Recognition** – The handwriting recognition software reads the text written on paper by a pen or on screen by a stylus. It can recognize the shapes of the letters and convert it into editable text.
- **Intelligent Robots** – Robots are able to perform the tasks given by a human. They have sensors to detect physical data from the real world such as light, heat, temperature, movement, sound, bump, and pressure. In addition, they are capable of learning from their mistakes and they can adapt to the new environment.

What is Intelligence?

- The ability of a system to calculate, reason, perceive relationships and analogies, learn from experience, store and retrieve information from memory, solve problems, comprehend complex ideas, use natural language fluently, classify, generalize, and adapt new situations.

Types of Intelligence

- As described by Howard Gardner, an American developmental psychologist, the Intelligence comes in multifold –

Intelligence	Description	Example
Linguistic intelligence	The ability to speak, recognize, and use mechanisms of phonology (speech sounds), syntax (grammar), and semantics (meaning).	Narrators, Orators

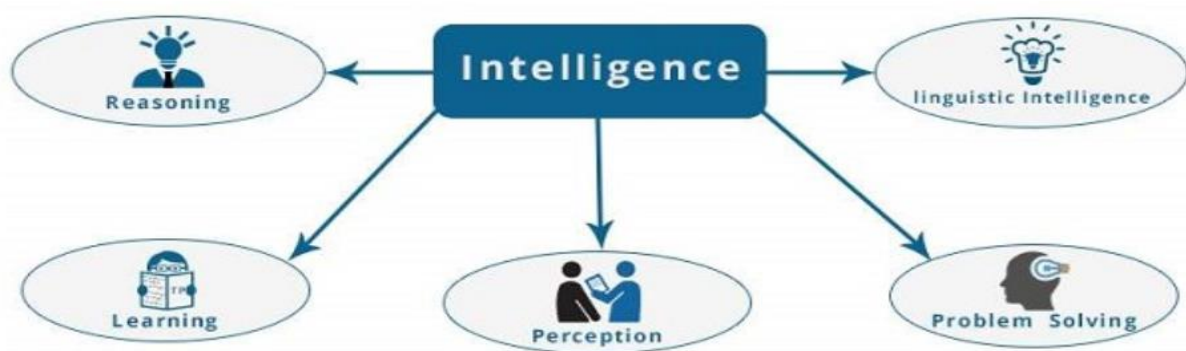
- **Musical intelligence** - The ability to create, communicate with, and understand meanings made of sound, understanding of pitch, rhythm.

- **Logical-mathematical intelligence** - The ability of use and understand relationships in the absence of action or objects. Understanding complex and abstract ideas.
- **Spatial intelligence** - The ability to perceive visual or spatial information, change it, and re-create visual images without reference to the objects, construct 3D images, and to move and rotate them.

What is Intelligence Composed of?

The intelligence is intangible. It is composed of –

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



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

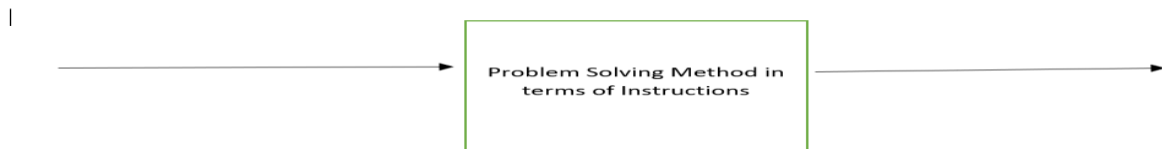
Inductive Reasoning	Deductive Reasoning
It conducts specific observations to makes broad general statements.	It starts with a general statement and examines the possibilities to reach a specific, logical conclusion.
Even if all of the premises are true in a statement, inductive reasoning allows for the conclusion to be false.	If something is true of a class of things in general, it is also true for all members of that class.
Example – "Nita is a teacher. Nita is studious. Therefore, All teachers are studious."	Example – "All women of age above 60 years are grandmothers. Shalini is 65 years. Therefore, Shalini is a grandmother."

- **Learning** – It is the activity of gaining knowledge or skill by studying, practicing, being taught, or experiencing something. Learning enhances the awareness of the subjects of the study.
- The ability of learning is possessed by humans, some animals, and AI-enabled systems. Learning is categorized as –
 - **Auditory Learning** – It is learning by listening and hearing. For example, students listening to recorded audio lectures.
 - **Episodic Learning** – To learn by remembering sequences of events that one has witnessed or experienced. This is linear and orderly.
 - **Motor Learning** – It is learning by precise movement of muscles. For example, picking objects, Writing, etc.
 - **Observational Learning** – To learn by watching and imitating others. For example, child tries to learn by mimicking her parent.
 - **Perceptual Learning** – It is learning to recognize stimuli that one has seen before. For example, identifying and classifying objects and situations.
 - **Relational Learning** – It involves learning to differentiate among various stimuli on the basis of relational properties, rather than absolute properties. For Example, Adding ‘little less’ salt at the time of cooking potatoes that came up salty last time, when cooked with adding say a tablespoon of salt.
 - **Spatial Learning** – It is learning through visual stimuli such as images, colors, maps, etc. For Example, A person can create roadmap in mind before actually following the road.
 - **Stimulus-Response Learning** – It is learning to perform a particular behavior when a certain stimulus is present. For example, a dog raises its ear on hearing doorbell.
 - **Problem Solving** – It is the process in which one perceives and tries to arrive at a desired solution from a present situation by taking some path, which is blocked by known or unknown hurdles.
 - -Problem solving also includes **decision making**, which is the process of selecting the best suitable alternative out of multiple alternatives to reach the desired goal are available.
 - **Perception** – It is the process of acquiring, interpreting, selecting, and organizing sensory information.

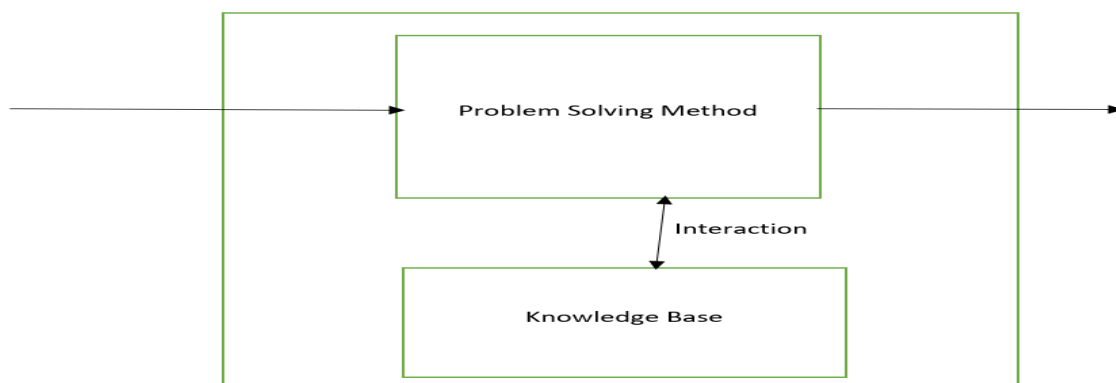
- Perception presumes **sensing**. In humans, perception is aided by sensory organs. In the domain of AI, perception mechanism puts the data acquired by the sensors together in a meaningful manner.
- **Linguistic Intelligence** – It is one's ability to use, comprehend, speak, and write the verbal and written language. It is important in interpersonal communication.

Difference between Human and Machine Intelligence

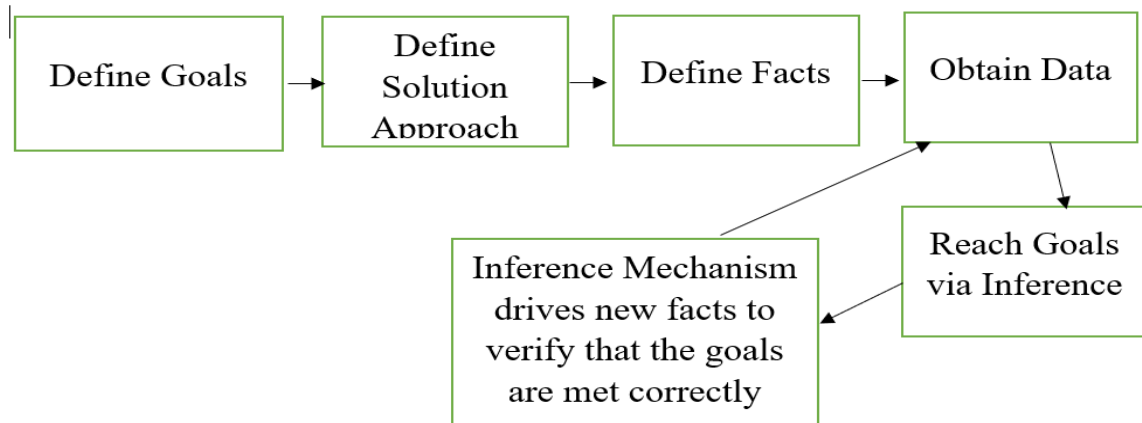
- Humans perceive by patterns whereas the machines perceive by set of rules and data.
- Humans store and recall information by patterns, machines do it by searching algorithms. For example, the number 40404040 is easy to remember, store, and recall as its pattern is simple.
- Humans can figure out the complete object even if some part of it is missing or distorted; whereas the machines cannot do it correctly.
- Comparison between architecture of AI and other conventional programs



- Architecture of AI



Components of AI



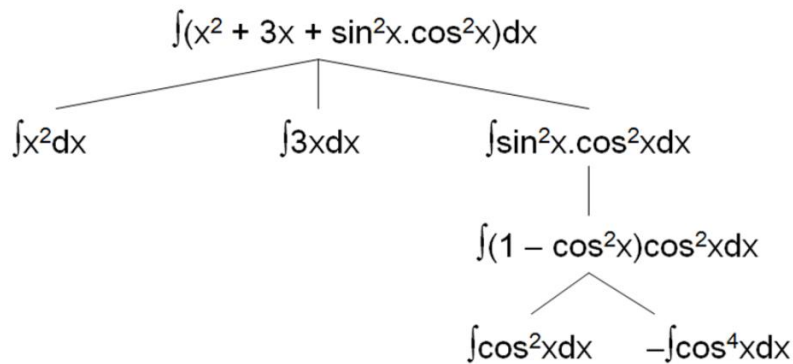
Problem Characteristics in Artificial Intelligence

- Since artificial intelligence (AI) is mainly related to the **search process**, it is important to have some methodology to choose the best possible solution.
- To choose an appropriate method for a particular problem first we need to categorize the problem based on the following characteristics.
 1. Is the problem decomposable into small sub-problems which are easy to solve?
 2. Can solution steps be ignored or undone?
 3. Is the universe of the problem is predictable?
 4. Is a good solution to the problem is absolute or relative?
 5. Is the solution to the problem a state or a path?
 6. What is the role of knowledge in solving a problem using artificial intelligence?
 7. Does the task of solving a problem require human interaction?

1. Is the problem decomposable into small sub-problems which are easy to solve?

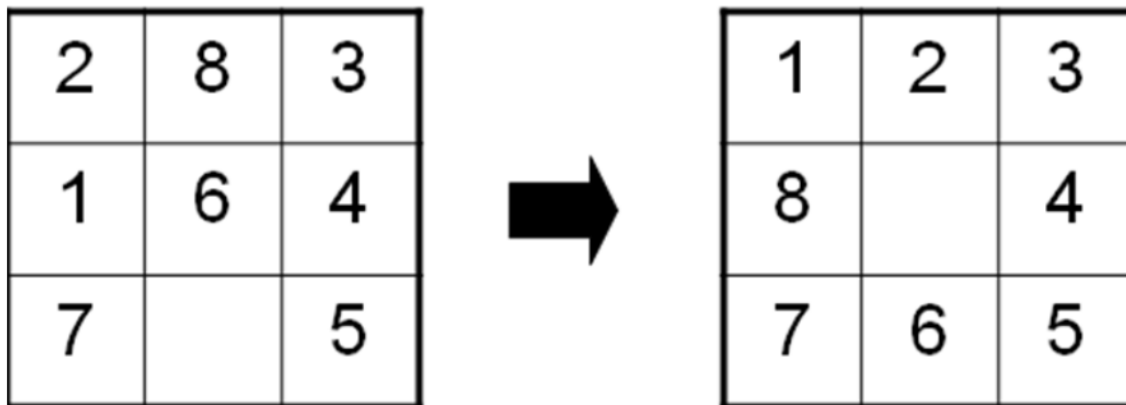
- Can the problem be broken down into smaller problems to be solved independently?
- The decomposable problem can be solved easily.

Example: In this case, the problem is divided into smaller problems. The smaller problems are solved independently. Finally, the result is merged to get the final result.



2. Can solution steps be ignored or undone?

- In the Theorem Proving problem, a lemma that has been proved can be ignored for the next steps.
- Such problems are called **Ignorable** problems.
- In the 8-Puzzle, Moves can be undone and backtracked.
- Such problems are called **Recoverable** problems.



- In Playing Chess, moves can be retracted.
- Such problems are called **Irrecoverable** problems.
- **Ignorable** problems can be solved using a simple control structure that never backtracks.
- **Recoverable** problems can be solved using backtracking.
- **Irrecoverable** problems can be solved by recoverable style methods via planning.

3. Is the universe of the problem is predictable?

- In Playing Bridge, We cannot know exactly where all the cards are or what the other players will do on their turns.

Uncertain outcome!

- For **certain-outcome problems**, planning can be used to generate a sequence of operators that is guaranteed to lead to a solution.
- For **uncertain-outcome problems**, a sequence of generated operators can only have a good probability of leading to a solution. Plan revision is made as the plan is carried out and the necessary feedback is provided.

4. Is a good solution to the problem is absolute or relative?

- The Travelling Salesman Problem, we have to try all paths to find the shortest one.
- Any path problem can be solved using heuristics that suggest good paths to explore.
- For best-path problems, a much more exhaustive search will be performed.

5. Is the solution to the problem a state or a path

- The Water Jug Problem, the path that leads to the goal must be reported.
- A path-solution problem can be reformulated as a state-solution problem by describing a state as a partial path to a solution. The question is whether that is natural or not.

6. What is the role of knowledge in solving a problem using artificial intelligence?

Playing Chess

- Consider again the problem of playing chess. Suppose you had unlimited computing power available.
- How much knowledge would be required by a perfect program?
- The answer to this question is very little— just the rules for determining legal moves and some simple control mechanism that implements an appropriate search procedure.
- Additional knowledge about such things as good strategy and tactics could of course help considerably to constrain the search and speed up the execution of the program. Knowledge is important only to constrain the search for a solution.

7. Does the task of solving a problem require human interaction?

- we are building programs that require intermediate interaction with people, both to provide additional input to the program and to provide additional reassurance to the user.
- The **solitary problem**, in which there is no intermediate communication and no demand for an explanation of the reasoning process.

- The **conversational problem**, in which intermediate communication is to provide either additional assistance to the computer or additional information to the user.

THE MR. WOLF, MR. GOAT AND MR. CABBAGE PROBLEM

Description:

- You are on the beginning of a bridge with Mr. Wolf, Mr. Goat and Mr. Cabbage.
- It's dark and the group has only one flash light. Your task is to get everyone to the other side.

Restrictions:

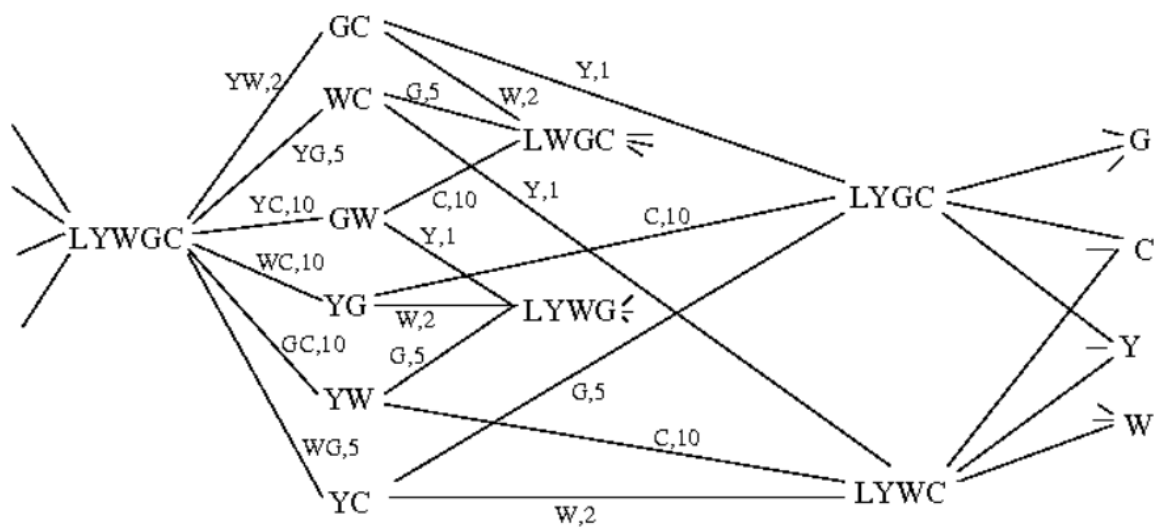
- Atmost two people can be on the bridge at the same time.
- People need the flash light to cross the bridge.

Cost:

- You can cross the bridge in 1 minute.
- Mr. Wolf needs 2 minutes.
- Mr. Goat needs 5 minutes.
- Mr. Cabbage needs 10 minutes.
- If two people cross the bridge together, they need as much time as the slowest of them.

Model

Model (part)



The wolf-goat-cabbage problem

Description

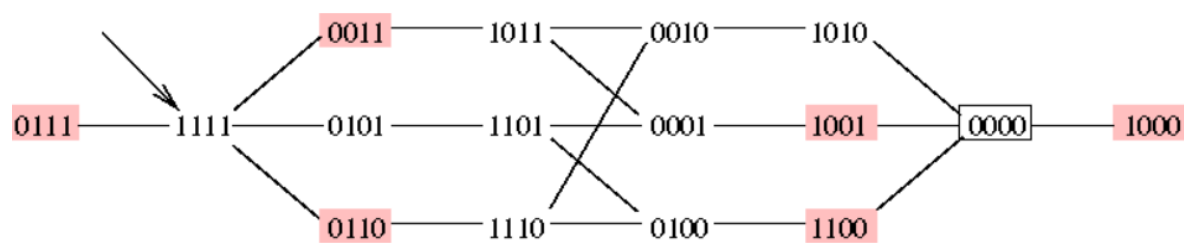
- You are on the bank of a river with a boat, a cabbage, a goat, and a wolf.
- Your task is to get everything to the other side.

Restrictions:

1. only you can handle the boat
2. when you're in the boat, there is only space for one more item
3. you can't leave the goat alone with the wolf, nor with the cabbage (or something will be eaten)

Model

- Model the *state* by 4 bits (for boat, cabbage, goat and wolf).
- A 1 means that the item is on this bank, a 0 means it's on the other bank.



Notes

Notes

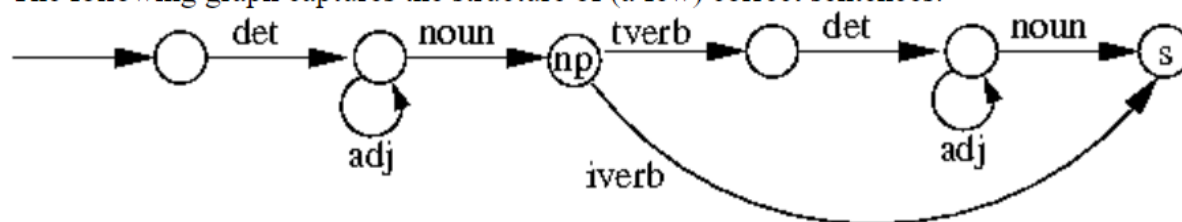
- Restriction 1 is in the state model: "you" are not modeled separate from the boat.
- Restriction 2 determines the possible moves. All moves can be made in both directions. (This is special for this problem. Normally, moves are *directed* from one state to another.)
- Restriction 3 is in the shaded states: these are not allowed.
- Early work in the AI focused on formal tasks, such as game playing and theorem proving.
- AI problems are appearing to have a very little in common but they are very hard.
- There must be some techniques that are appropriate for the solution of a variety of these problems.

Natural language processing

- Part of natural language processing is to decide which roles the words of a sentence have. Since a word can have different roles and meanings, this is a search problem.

- There are many ways to tackle these problems. The following is simple, but not very powerful.

The following graph captures the structure of (a few) correct sentences.



We can combine this with a lexicon

word	word class	meaning
the	determiner	
old	adjective	of high age
old	noun	people of high age
man	noun	male person
man	noun	mankind

man	transitive verb	be the crew of
ship	noun	boat
ship	transitive verb	send
rows	transitive verb	propel by rowing
rows	noun	lines in a table

The monkey and the banana

- The purpose of this example is to show the use of variables.

Description

A monkey enters a room via the door. In the room, near the window, is a box. In the middle of the room hangs a banana from the ceiling. The monkey wants to grasp the banana, and can do so after climbing on the box in the middle of the room.

- **States**

For each state, we need to record:

- the position of the monkey (door, window, middle, ...)
- the position of the box
- if the monkey is on the box
- if the monkey has the banana

The initial state is (door, window, no, no).

The set of goal states is (*, *, *, yes).

- **Moves**

walk(P): from (M, B, no, H) to (P, B, no, H).

push(P): from (M, M, no, H) to (P, P, no, H).

climb: from (M, M, no, H) to (M, M, yes, H).

grasp: from (middle, B, yes, no) to (middle, B, yes, yes).

- **State space**

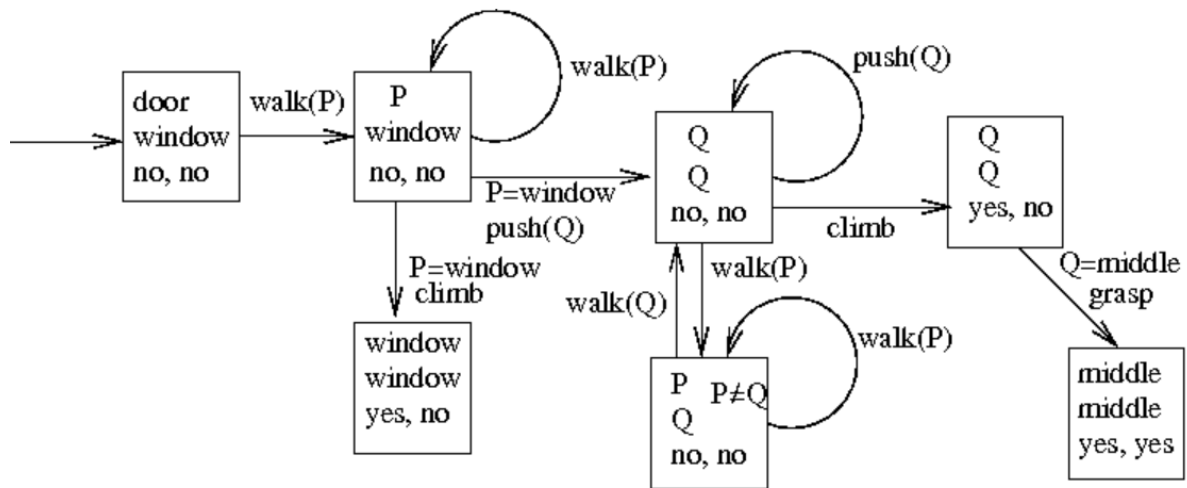
Without variables, the state space and search space can be very large (how many positions are there?).

With variables, we can represent the reachable part as follows.

Approach to Solve the Problem

- Normally, any real-world problem can be modeled as a set of states and actions like this:



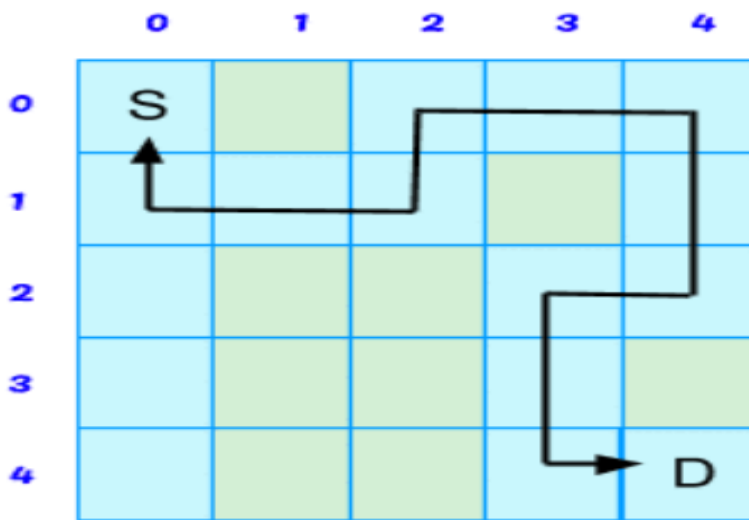


State Space Search

- In AI a state space consists of the following elements,
 1. A(possibly infinite) set of states
 - a) Out of the possible states, one state represents start state that is the initial state of the problem.
 - b) Each state represents some configuration so that it is reachable from the start state.
 - c) Out of the possible states, some states may be goal states(solutions)
 2. A set of rules,
 - a) Applying a rule to the current state, transforms it to another or a new state in the state space
 - b) All operators may not be applicable to all states in the state space
- State spaces are used extensively in Artificial Intelligence (AI) to represent and solve problems.

State Space Search Examples:

- **Example 1. Maze**




A maze problem can be represented as a state-space


- Each state represents “where you are” that is the current position in the maze
- The start state or initial state represents your starting position
- The goal state represents the exit from the maze
- Rules (for a rectangular maze) are: move north, move south, move east, and move west
- Each rule takes you to a new state (maze location)
- Rules may not always apply, because of walls in the maze

Example 2. The 15 Puzzle

Start state:

3	10	13	7
9	14	6	1
4		15	2
11	8	5	12

Goal state:

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

The start state is some (almost) random configuration of the tiles and the goal state is as shown.

State Space Search Rules are

- Move empty space up
- Move empty space down
- Move empty space right
- Move empty space left
- These Rules apply if empty space is not against the edge.

1. First, select some way to represent states in the given problem in an unambiguous way.
2. Next, formulate all actions or operators that can be performed in states, including their preconditions and effects.
 - *Actions or operates are called **PRODUCTION RULES***
3. Represent the initial state or states of the problem.
4. Formulate precisely when a state satisfies the goal of our problem.
5. Activate the production rules on the initial state and its descendants, until a goal state is reached.

State space representation of a water jug problem.

- We are given two jugs, a 4-gallon one and 3-gallon one. Neither has any measuring marked on it. There is a pump, which can be used to fill the jugs with water. How can we get exactly 2 gallons of water into 4-gallon jug?

- The state space for this problem can be described as the set of ordered pairs of integers (X, Y) such that $X = 0, 1, 2, 3$ or 4 and $Y = 0, 1, 2$ or 3 ;
- X is the number of gallons of water in the 4-gallon jug and Y the quantity of water in the 3-gallon jug.
- The start state is $(0, 0)$ and the goal state is $(2, n)$ for any value of n , as the problem does not specify how many gallons need to be filled in the 3-gallon jug $(0, 1, 2, 3)$.
- So the problem has one initial state and many goal states. Some problems may have many initial states and one or many goal states.

1.	(X, Y) if $X < 4 \rightarrow (4, Y)$	Fill the 4-gallon jug
2.	(X, Y) if $Y < 3 \rightarrow (X, 3)$	Fill the 3-gallon jug
3.	(X, Y) if $X = d \ \& \ d > 0 \rightarrow (X-d, Y)$	Pour some water out of the 4-gallon jug
4.	(X, Y) if $Y = d \ \& \ d > 0 \rightarrow (X, Y-d)$	Pour some water out of 3-gallon jug
5.	(X, Y) if $X > 0 \rightarrow (0, Y)$	Empty the 4-gallon jug on the ground
6.	(X, Y) if $Y > 0 \rightarrow (X, 0)$	Empty the 3-gallon jug on the ground
7.	(X, Y) if $X + Y \leq 4$ and $Y > 0 \rightarrow 4, (Y - (4 - X))$	Pour water from the 3-gallon jug into the 4-gallon jug until the gallon jug is full.
8.	(X, Y) if $X + Y \geq 3$ and $X > 0 \rightarrow (X - (3 - Y), 3)$	Pour water from the 4-gallon jug into the 3-gallon jug until the 3-gallon jug is full.
9.	(X, Y) if $X + Y \leq 4$ and $Y > 0 \rightarrow (X + Y, 0)$	Pour all the water from the 3-gallon jug into the 4-gallon jug
10.	(X, Y) if $X + Y \leq 3$ and $X > 0 \rightarrow (0, X + Y)$	Pour all the water from the 4-gallon jug into the 3-gallon jug
11.	$(0, 2) \rightarrow (2, 0)$	Pour the 2-gallons water from 3-gallon jug into the 4-gallon jug
12.	$(2, Y) \rightarrow (0, Y)$	Empty the 2-gallons in the 4-gallon jug on the ground.

- In order to describe the operators completely here are some assumptions, not mentioned, in the problem state.
- We can fill a jug from the pump.
- We can pour water out a jug, onto the ground.
- We can pour water out of one jug into the other.
- No other measuring devices are available.
- Having a control structure which loops through a simple cycle in which some rule whose left side matches the current state is chosen, the appropriate change to the state is made as described in the corresponding right side and the resulting state is checked to see if it corresponds to a goal state.

- There are several sequences of operators which will solve the problem, two such sequences are shown

Water in 4-gallon jug (X)	Water in 3-gallon jug (Y)	Rule applied
0	0	
0	3	2
3	0	9
3	3	2
4	2	7
0	2	5 or 12
2	0	9 or 11

Fig. 2.4 (a). A solution to water jug problem.

X	Y	Rule applied (Control strategy)
0	0	
4	0	I-
1	3	8
1	0	6
0	1	10
4	1	1
2	3	8

Fig. 2.4 (b). 2nd solution to water jug problem.

PRODUCTION SYSTEMS

- Production systems provide appropriate structures for performing and describing search processes.
- A production system has four basic components as enumerated below.
 1. A set of rules each consisting of a left side that determines the applicability of the rule and a right side that describes the operation to be performed if the rule is applied.
 2. A database of current facts established during the process of inference.
 3. A control strategy that specifies the order in which the rules will be compared with facts in the database and also specifies how to resolve conflicts in selection of several rules or selection of more facts.
 4. A rule firing module.

The production rules operate on the knowledge database.

- Each rule has a precondition—that is, either satisfied or not by the knowledge database.
- If the precondition is satisfied, the rule can be applied. Application of the rule changes the knowledge database.

- The control system chooses which applicable rule should be applied and ceases computation when a termination condition on the knowledge database is satisfied.

Example: Eight puzzle (8-Puzzle)

- The 8-puzzle is a 3×3 array containing eight square pieces, numbered 1 through 8, and one empty space.
- A piece can be moved horizontally or vertically into the empty space, in effect exchanging the positions of the piece and the empty space.
- There are four possible moves, UP (move the blank space up), DOWN, LEFT and RIGHT.

The aim of the game is to make a sequence of moves that will convert the board from the start state into the goal state:

2	3	4
8	6	2
7		5

Initial State

1	2	3
8		4
7	6	5

Goal State

This example can be solved by the operator sequence UP, RIGHT, UP, LEFT, DOWN

Example: Missionaries and Cannibals

- The Missionaries and Cannibals problem illustrates the use of state space search for planning under constraints:
- Three missionaries and three cannibals wish to cross a river using a two person boat.
- If at any time the cannibals outnumber the missionaries on either side of the river, they will eat the missionaries.
- How can a sequence of boat trips be performed that will get everyone to the other side of the river without any missionaries being eaten?

State representation:

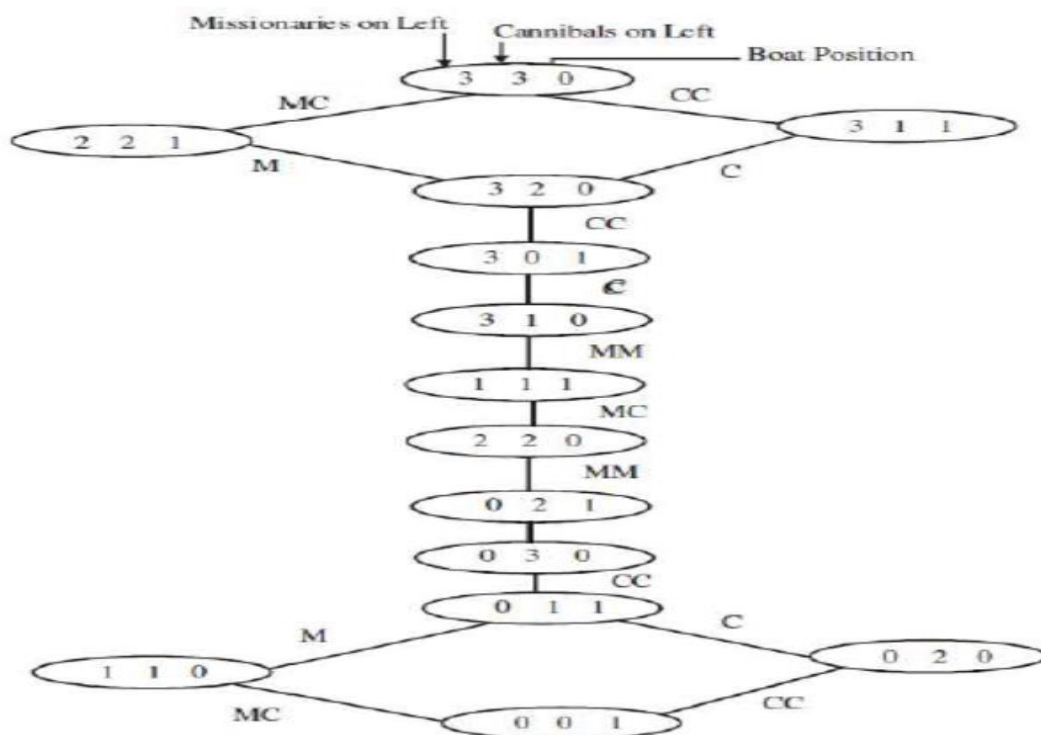
1. BOAT position: original (T) or final (NIL) side of the river.
2. Number of Missionaries and Cannibals on the original side of the river.

3. Start is (T 3 3); Goal is (NIL 0 0).

Operators:

(MM 2 0)	Two Missionaries cross the river.
(MC 1 1)	One Missionary and one Cannibal.
(CC 0 2)	Two Cannibals.
(M 1 0)	One Missionary.
(C 0 1)	One Cannibal.

Missionaries/Cannibals Search Graph



CHARACTERISTICS OF PRODUCTION SYSTEMS

- Production systems provide us with good ways of describing the operations that can be performed in a search for a solution to a problem.
- At this time, two questions may arise:

1. Can production systems be described by a set of characteristics? And how can they be easily implemented?

2. What relationships are there between the problem types and the types of production systems well suited for solving the problems?

- To answer these questions, first consider the following definitions of classes of production systems:
- A monotonic production system is a production system in which the application of a rule never prevents the later application of another rule that could also have been applied at the time the first rule was selected.
- A non-monotonic production system is one in which this is not true.
- A partially communicative production system is a production system with the property that if the application of a particular sequence of rules transforms state P into state Q, then any combination of those rules that is allowable also transforms state P into state Q.
- A commutative production system is a production system that is both monotonic and partially commutative.

Issues in the Design of Search Programs

- Each search process can be considered to be a tree traversal.

The following issues arise when searching:

- The tree can be searched forward from the initial node to the goal state or backwards from the goal state to the initial state.
- To select applicable rules, it is critical to have an efficient procedure for matching rules against states.
- How to represent each node of the search process? This is the knowledge representation problem or the frame problem. In games, an array suffices; in other problems, more complex data structures are needed.

HEURISTIC SEARCH TECHNIQUES:

Search Algorithms

- Many traditional search algorithms are used in AI applications. For complex problems, the traditional algorithms are unable to find the solutions within some practical time and space limits. Consequently, many special techniques are developed, using *heuristic functions*.

- The algorithms that use *heuristic functions* are called ***heuristic algorithms***.
- Heuristic algorithms are not really intelligent; they appear to be intelligent because they achieve better performance.
- Heuristic algorithms are more efficient because they take advantage of feedback from the data to direct the search path.
- ***Uninformed search algorithms*** or *Brute-force algorithms*, search through the search space all possible candidates for the solution checking whether each candidate satisfies the problem's statement.
- ***Informed search algorithms*** use heuristic functions that are specific to the problem, apply them to guide the search through the search space to try to reduce the amount of time spent in searching.

Some prominent intelligent search algorithms are stated below:

1. Generate and Test Search

2. Best-first Search

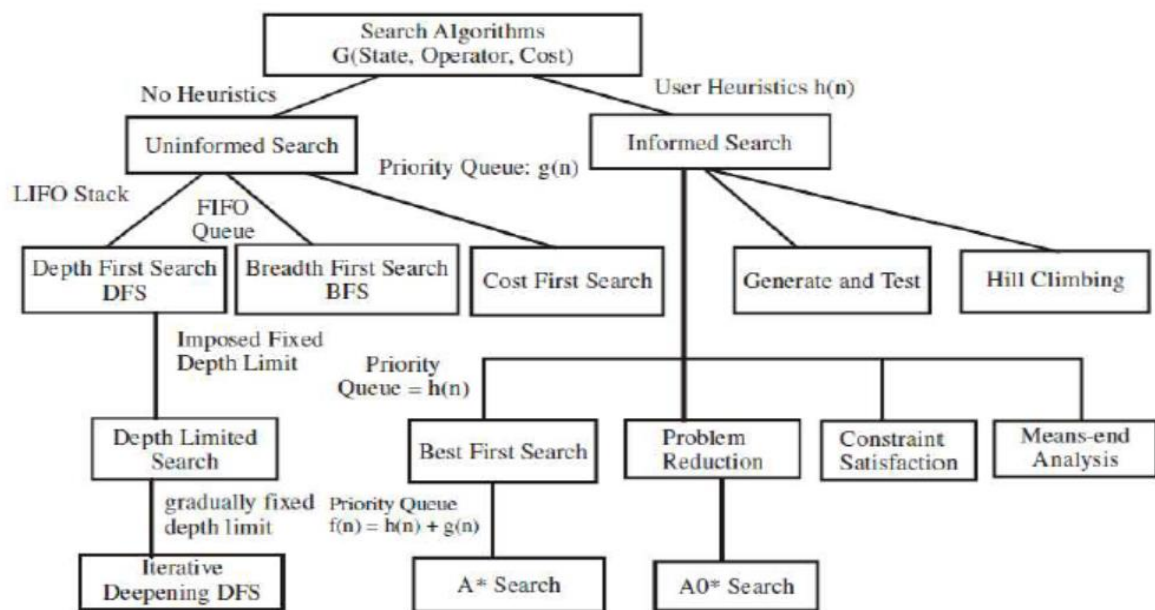
3. Greedy Search

4. A* Search

5. Constraint Search

6. Means-ends analysis

- There are some more algorithms. They are either improvements or combinations of these.
- **Hierarchical Representation of Search Algorithms:** A Hierarchical representation of most search algorithms is illustrated below. The representation begins with two types of search:
- **Uninformed Search:** Also called blind, exhaustive or brute-force search, it uses no information about the problem to guide the search and therefore may not be very efficient.
- **Informed Search:** Also called heuristic or intelligent search, this uses information about the problem to guide the search—usually guesses the distance to a goal state and is therefore efficient, but the search may not be always possible.



Breadth-first search

- A Search strategy, in which the highest layer of a decision tree is searched completely before proceeding to the next layer is called *Breadth-first search (BFS)*.
- In this strategy, no viable solutions are omitted and therefore it is guaranteed that an optimal solution is found.
- This strategy is often not feasible when the search space is large.

Algorithm

1. Create a variable called LIST and set it to be the starting state.
2. Loop until a goal state is found or LIST is empty, Do
 - a. Remove the first element from the LIST and call it E. If the LIST is empty, quit.
 - b. For every path each rule can match the state E, Do
 - (i) Apply the rule to generate a new state.
 - (ii) If the new state is a goal state, quit and return this state.
 - (iii) Otherwise, add the new state to the end of LIST.

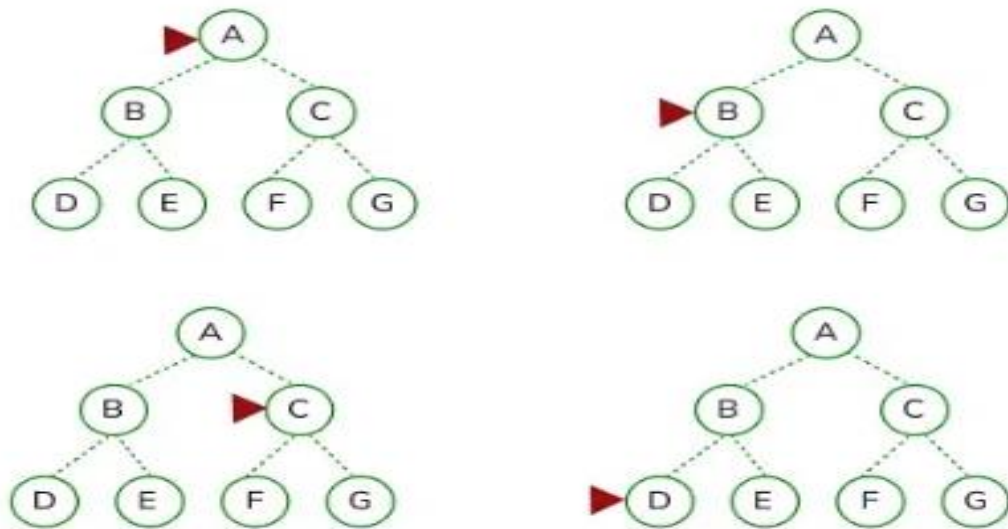
Advantages

1. Guaranteed to find an optimal solution (in terms of shortest number of steps to reach the goal).
2. Can always find a goal node if one exists (complete).

Disadvantages

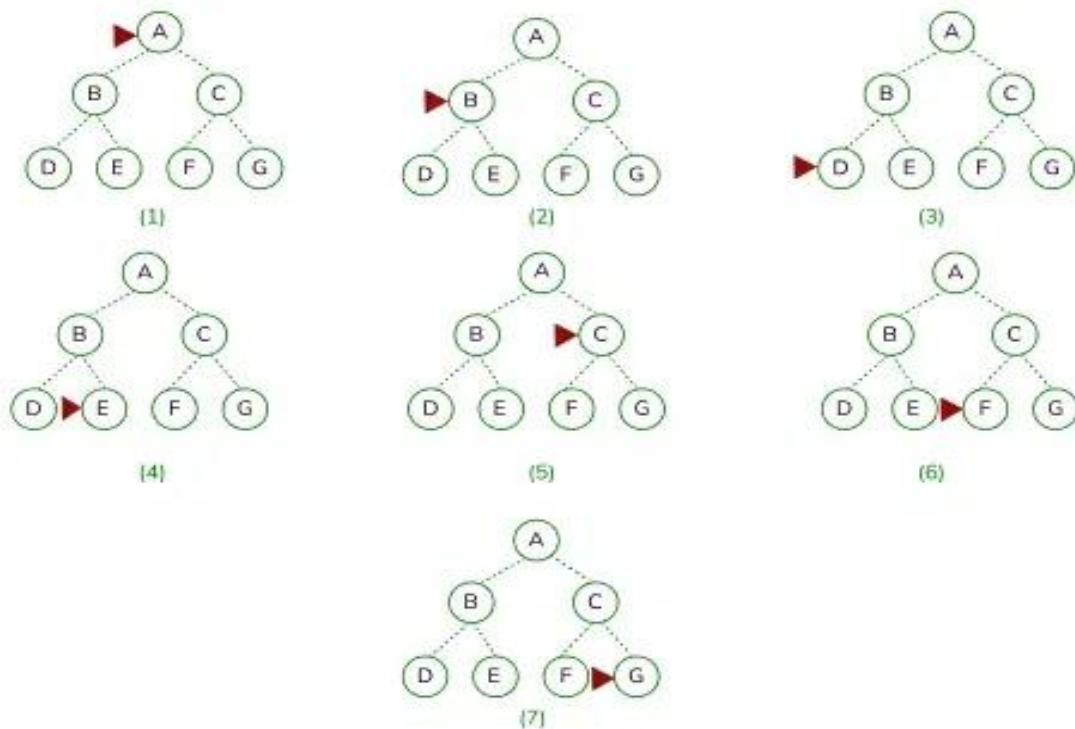
1. High storage requirement: *exponential* with tree depth.

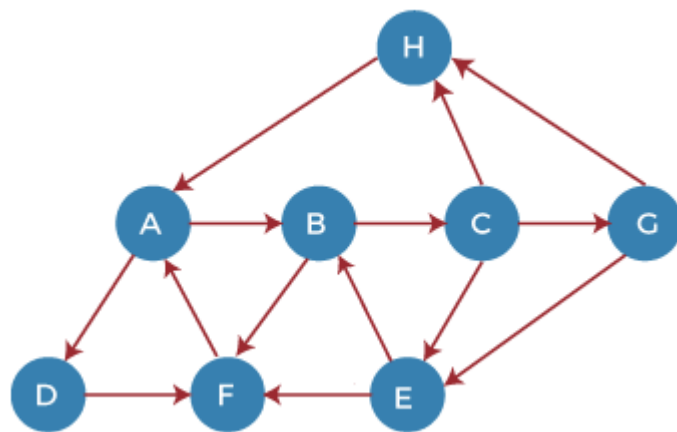
The graph structure of BFS allows to work as follows:



Depth First Search

- Depth-first search (DFS) explores a graph by selecting a path and traversing it as deeply as possible before backtracking.





Adjacency Lists

A : B, D
 B : C, F
 C : E, G, H
 G : E, H
 E : B, F
 F : A
 D : F
 H : A

DFS(G,v) (v is the vertex where the search starts)

Stack $S := \{\}$; (start with an empty stack)

for each vertex u, set visited[u] := **false**;

push S, v;

while (S is not empty) **do**

u := pop S;

if (not visited[u]) then

visited[u] := **true**;

for each unvisited neighbour w of u

push S, w;

end **if**

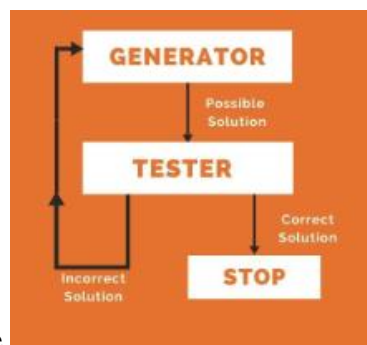
end **while**

END DFS()

Heuristic Search Techniques

Generate and Test

- Heuristic technique, DFS with backtracking
- Steps
 - Generate the possible solution
 - Test to see if this is a actual solution
 - If a solution is found, Quit. Otherwise go to step -1
- Properties of Good generators
 - Complete
 - Non Redundant
 - Informed



- Example
- Three number pin, Every number having two digits
- 00 00 00 , find the possible answer

Brute force method:

00 00 00

00 00 01

Possibility- $100 \times 100 \times 100 = 1M$

- Let say for every minute 5 possible solutions we get
- 1minute=5 , 1 hour=5*60=300 (solution we get in 10 weeks)

Hueristic – Domain knowlegde

Digits are prime no's-

- **Possibility- $25*25*25=15625$**

Best First Search

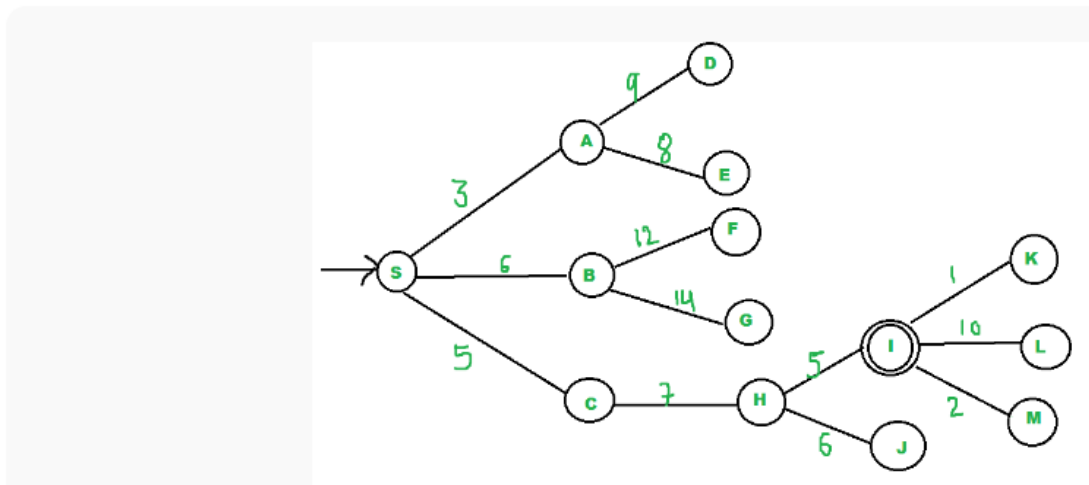
- If we consider searching as a form of traversal in a graph, an uninformed search algorithm would blindly traverse to the next node in a given manner without considering the cost associated with that step.
- An informed search, like BFS, on the other hand, would use an evaluation function to decide which among the various available nodes is the most promising (or 'BEST') before traversing to that node.
- BFS uses the concept of a Priority queue and heuristic search. To search the graph space, the BFS method uses two lists for tracking the traversal.
- An 'Open' list that keeps track of the current 'immediate' nodes available for traversal and a 'CLOSED' list that keeps track of the nodes already traversed.

Best First Search Algorithm

- Step 1 : Create a priorityQueue pqueue.
- Step 2 : insert 'start' in pqueue : `pqueue.insert(start)`
- Step 3 : delete all elements of pqueue one by one.
 - Step 3.1 : if, the element is goal . Exit.
 - Step 3.2 : else, traverse neighbours and mark the node examined.
- Step 4 : End.
- This algorithm will traverse the shortest path first in the queue. The time complexity of the algorithm is given by $O(n*\log n)$.
- The idea of **Best First Search** is to use an evaluation function to decide which adjacent is most promising and then explore.
- We use a priority queue or heap to store the costs of nodes that have the lowest evaluation function value. So the implementation is a variation of BFS, we just need to change Queue to PriorityQueue.

```
// Pseudocode for Best First Search
Best-First-Search(Graph g, Node start)
  1) Create an empty PriorityQueue
     PriorityQueue pq;
  2) Insert "start" in pq.
     pq.insert(start)
  3) Until PriorityQueue is empty
     u = PriorityQueue.DeleteMin
     If u is the goal
       Exit
     Else
       Foreach neighbor v of u
         If v "Unvisited"
           Mark v "Visited"
           pq.insert(v)
       Mark u "Examined"
End procedure
```

Let us consider the below example:



- We start from source “S” and search for goal “I” using given costs and Best First search.
- pq initially contains S
 - We remove s from and process unvisited neighbors of S to pq.
 - pq now contains {A, C, B} (C is put before B because C has lesser cost)
- We remove A from pq and process unvisited neighbors of A to pq.

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

Introduction to Hill Climbing

- Hill Climbing is a heuristic search used for mathematical optimization problems in the field of Artificial Intelligence.
- Given a large set of inputs and a good heuristic function, it tries to find a sufficiently good solution to the problem.
- This solution may not be the global optimal maximum
- In the above definition, **mathematical optimization problems** imply that hill-climbing solves the problems where we need to maximize or minimize a given real function by choosing values from the given inputs.
- Example-Travelling salesman problem where we need to minimize the distance traveled by the salesman.
- 'Heuristic search' means that this search algorithm may not find the optimal solution to the problem. However, it will give a good solution in a **reasonable time**.
- A **heuristic function** is a function that will rank all the possible alternatives at any branching step in the search algorithm based on the available information.
- It helps the algorithm to select the best route out of possible routes.

Hill Climbing Algorithm

- Hill climbing algorithm is a local search algorithm which continuously moves in the direction of increasing elevation/value to find the peak of the mountain or best solution to the problem.
- Hill climbing algorithm is a technique which is used for optimizing the mathematical problems.

- It is also called greedy local search as it only looks to its good immediate neighbor state and not beyond that.
- A node of hill climbing algorithm has two components which are state and value.
- Hill Climbing is mostly used when a good heuristic is available.
- In this algorithm, we don't need to maintain and handle the search tree or graph as it only keeps a single current state

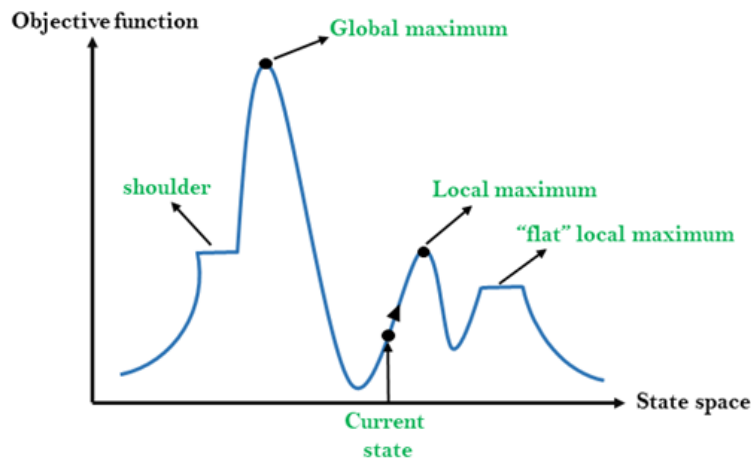
Features of Hill Climbing:

Following are some main features of Hill Climbing Algorithm:

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

State-space Diagram for Hill Climbing:

- The state-space landscape is a graphical representation of the hill-climbing algorithm.
- On Y-axis we have taken the function which can be an objective function or cost function, and state-space on the x-axis.
- If the function on Y-axis is cost then, the goal of search is to find the global minimum and local minimum.
- If the function of Y-axis is Objective function, then the goal of the search is to find the global maximum and local maximum.
- A **local minimum** is a point in the search space where the value of the objective function is lower than at all neighboring points, but not necessarily the lowest possible value in the entire search space.
- A **global minimum** is the absolute lowest point in the entire search space. It is the best possible solution to the problem.



- **Local Maximum:** Local maximum is a state which is better than its neighbor states, but there is also another state which is higher than it.
- **Global Maximum:** Global maximum is the best possible state of state space landscape. It has the highest value of objective function.
- **Current state:** It is a state in a landscape diagram where an agent is currently present.
- **Flat local maximum:** It is a flat space in the landscape where all the neighbor states of current states have the same value.
- **Shoulder:** It is a plateau region which has an uphill edge.

Case 1: Y-axis Represents Cost

- **Goal:** Minimize the cost.
- **Global Minimum:** This is the lowest point in the entire curve. Achieving this point means you've found the best (least costly) solution possible.
- **Local Minimum:** These are the lower points in the curve that are better than their immediate surroundings but not necessarily the best overall. The hill climbing algorithm might stop at one of these points because it only looks at nearby points.
- In this case, the search algorithm (like hill climbing) aims to descend the curve (reduce the cost) and ideally should reach the global minimum. However, it might get stuck in one of the local minima if it doesn't have a mechanism to escape it.

Case 2: Y-axis Represents an Objective Function

- **Goal:** Maximize the objective function.
- **Global Maximum:** This is the highest point in the entire curve. Achieving this point means you've found the best (highest value) solution possible.

- **Local Maximum:** These are the higher points in the curve that are better than their immediate surroundings but not necessarily the best overall. Again, a search algorithm might stop at one of these points if it doesn't explore the entire space.

Types of Hill Climbing Algorithm:

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

1. Simple Hill Climbing:

- Simple hill climbing is the simplest way to implement a hill climbing algorithm.
- It only evaluates the neighbor node state at a time and selects the first one which optimizes current cost and set it as a current state.
- It only checks its one successor state, and if it finds better than the current state, then move else be in the same state. This algorithm has the following features:
- Less time consuming
- Less optimal solution and the solution is not guaranteed

Algorithm for Simple Hill Climbing:

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

2.Steepest-Ascent algorithm

The steepest-Ascent algorithm is a variation of simple hill climbing algorithm.

- This algorithm examines all the neighboring nodes of the current state and selects one neighbor node which is closest to the goal state.
- This algorithm consumes more time as it searches for multiple neighbors

Algorithm for Steepest-Ascent hill climbing:

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

3. Stochastic hill climbing:

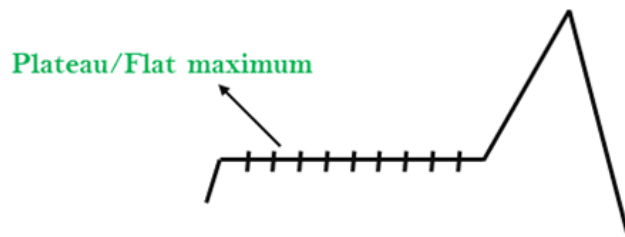
- Stochastic hill climbing does not examine for all its neighbor before moving. Rather, this search algorithm selects one neighbor node at random and decides whether to choose it as a current state or examine another state.

Problems in Hill Climbing Algorithm:

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



- **Plateau:** A plateau is the flat area of the search space in which all the neighbor states of the current state contains the same value. A hill-climbing search might be lost in the plateau area.



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

