# Introduction to AI

## Chapter 1: Introduction

#### What's AI

- Programs that behave like human externally.
- Porgram that operate like human internally.
- Discipline that systematizes and automates reasoning process to create machines that `act and think rationally like humans`.

#### Applications of AI

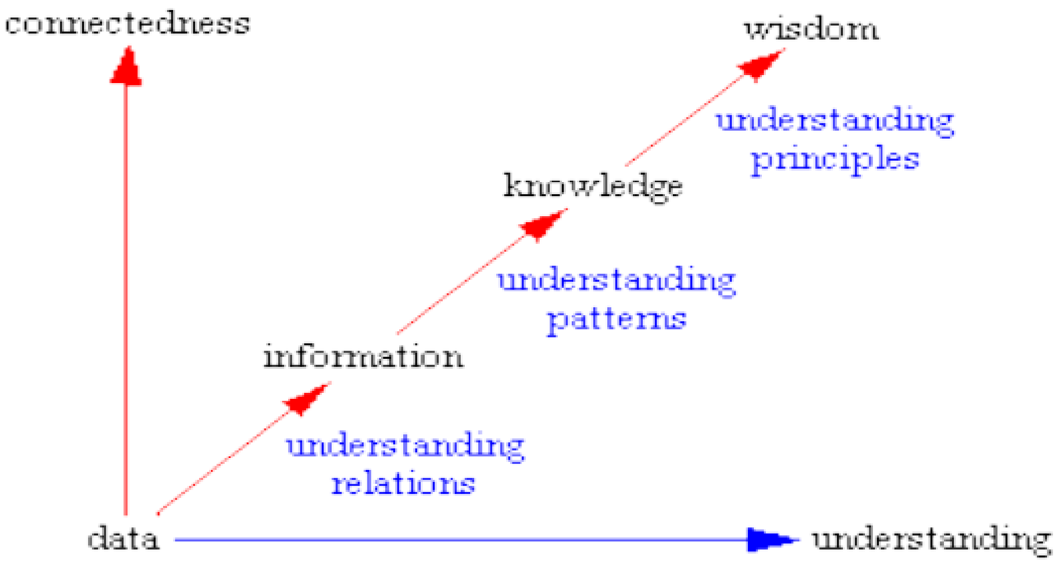
- Monitor trades
- Detect fraud
- Schedule shuttle loading
- A robot, web shopping program, a factory and a traffic control
- Software that gathers information about an environment and takes action based on that (Agent)

#### Data, Information, Knowledge and Wisdom

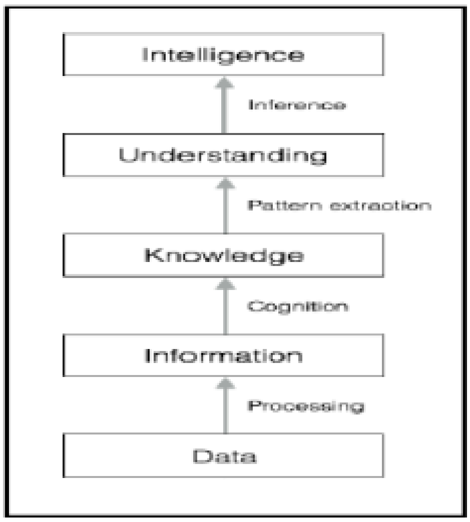
- Categories of the human mind:
    - **Data**: raw fact(symbols).
    - **Information**: data that are processed to be useful.
    - **Knowledge**: application of data and information (answers "how").
    - **Understanding**: appreciation of "why".
    - **Wisdom**: evaluated understanding.
- Data:
    - Exists in any form without any significance.
    - Doesn't have meaning by itself.
- Information:
    - Data that has been given meaning by the way of `relational connection`.
    - "Meaning" can be useful, but doesn't have to be.
    - $Infromation = Data + Context + Meaning$
- Knowledge:
    - The appropriate collection of information, such that it's intent is to be useful.
    - Deterministic process.
- Wisdom:

- Extrapolative, non-deterministic, and non-probabilistic process.
  - It calls upon all the previous levels of consciousness, and specifically upon special types of human programming `moral` , `ethical codes` , etc.
  - Since it's difficult to represent wisdom with computer, most people believe that computers don't have and will never have wisdom.

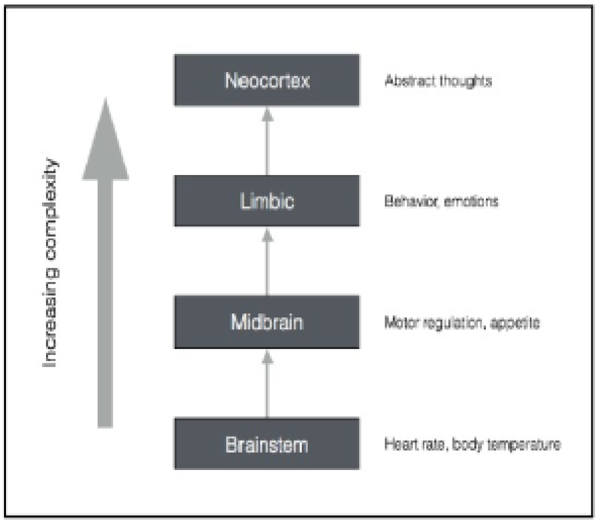
**Knowledge Hierarchy**




- The first four categories relate to the past, and only the fifth category, `wisdom` deals with the future because it incorporates `vision` and `design` .




How raw data gets converted to wisdom through various levels of processing




How our brain processes information

**Intelligence and Views of AI**

- Intelligence:
  - Capacity to learn and solve problems.
  - The ability to act rationally.
- Views/Perspectives of AI:

| | Human-like Intelligence | "Ideal" Intelligent/ Pure Rationality |
|---|---|---|
| Thought/ Reasoning | 2. Thinking humanly | 3. Thinking Rationally |
| Behavior/ Actions "behaviorism" | 1. Acting Humanly | 4. Acting Rationally |

- **Think like humans (The cognitive modeling approach)**:
  - We must have some way of determining how humans think and need to get inside the actual workings of human minds.
  - Thinking is also about unconscious tasks such as vision, speech understanding and reflex action.
- **Act like humans (The turing test approach)**:
  - Alan Turing test (Operational test for intelligent behavior: the imitation game).
  - Indistinguishability from undeniably intelligent entities(human beings).
  - Capabilities needed:
    - `Natural Language Processing` -> successful communication.
    - `Knowledge representation` -> store what it knows or hears.
    - `Automated reasoning` -> answer questions and make conclusions.
    - `Machine learning` -> adaptation, detect and extrapolate patterns.
    - `Computer vision` -> perceive objects
    - `Robotics` -> manipulate objects and move.
- **Think Rationally (The laws of thought)**:
  - Instead of thinking like a human, think rationally.
  - Uses syllogism (field under logic, a traditional and important branch of mathematics and computer science).
  - Problems:
    - It's not always possible to model thought as a set of rules; sometimes there is uncertainity.
    - The complexity of the problem may be too large to allow for solution.
- **Act Rationally (The rational agent approach)**:
  - An `agent` is an entity that perceives and acts.
  - `Relational agent` acts as to achieve best outcome.
  - Logical thinking is only one aspect of appropriate behavior.
  - Instead of insisting on how the program should think, it's better to insist on how the program should act.
  - Cares only about the final result(goal).

### Additional definitions of AI

- AI is an attempt of the reproduction of human reasoning and intelligent behavior by computational methods.
- The goal of AI is to create computer systems that perform tasks regarded as requiring intelligence when done by humans and behave "intelligently".

- AI involves modeling human and other animals activities, behaviour, and thoughts.
- AI involves building computer systems “do the right thing” in complex environments.
- Systems that build to act optimally given the limited information and computational resource avaliable.

**Foundations of AI**

- Philosophy
- Mathematics
- Economics
- Psychology
- Brain Science (Neuroscience)
- Physics
- Computer engineering
- Linguistics

**AI and Biochemistry**

- AI contributes to building an information processing model of human beings, just as Biochemistry contributes to building a model of human beings based on bio-molecular interactions.
- Both try to explain how a human being operates.
- Both also explore ways to avoid human imperfections.
- Both try to produce new useful technologies.
- Neither explains yet the true meaning of being human.

**Main areas of AI**

- Knowledge representation
- Search
- Planning
- Reasoning under uncertainity (probabilistic reasoning)
- Learning
- Agent architectures
- Robotics and perception
- Natural Language Processing

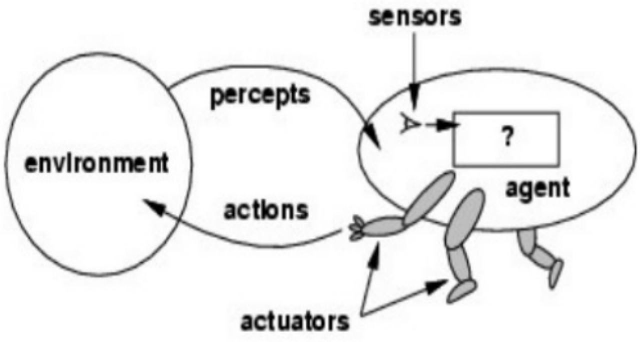
**History of AI**

- 1956: the name AI is coined.
- 1960’s: Search, games, formal logic and theorm proving.
- 1970’s: Robotics, perception, knowledge representation and expert systems.
- 1980’s: More expert systems, AI becomes an industry.
- 1990’a: Rational agents, probabilistic reasoning and machine learning.
- 2000’s: System integrated many AI methods, machine learning, reasoning under uncertainity and robotics.

# Chapter 2: Intelligent Agent

### What's an Agent?

- Anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators in order to meet its design objectives.




### Agent Function

- `Percept` refers to the agent's perceptual inputs at any given instant.
- `Percept sequence` is the complete history of everything the agent has perceived.
- An agent's behaviour is described by the Agent function that maps any given percept sequence to an action $[f : p* -> A]$.
- The agent program runs on the physical architecture to produce agent `function f`.
  - $Agent = architecture + program$
  - `Architecture` : the machinery that an agent executes on.
  - `Program` : an implementation of an agent function.

### How should Agents act?

- A `rational` agent is an agent that does the `right` thing for the perceived data from the environment.
- What is right is an ambiguous concept but we can consider the `right thing` as the one that makes the agent more `successful`.
- `Success` is also measured by using performance measure and a performance measure embodies the criterion for success of an agent's behavior.

### Performance Measures

- **Subjective Measure**:
  - How happy is the agent at the end of the action.
  - Agent should answer based on its opinion.
  - Some agents are unable to answer, some delude them selves, some over estimate and some under estimate their success.
- **Objective Measure**:
  - Needs standard to measure success.
  - Provides quantitative value of success measure of an agent.
  - Involves factors that affect performance and weight to each factors.

### Omniscience Agent Vs Rational Agent

| Omniscience | Rational |
|---|---|
| Knows the actual outcome of it's actions and act accordingly | Tries to achieve success from it's decision |
| Omniscience agent that act and think rationally never make a mistake | Rational agent could make a mistake because of unpredictable factors at the time of making decision |
| Maximizes actual performance | Maximizes expected performance |

### Autonomous Agent

- An agent is `autonomous` if its behavior is determined by its own experience (with ability to learn and adapt).
- Agent that `lacks autonomous`, if its actions are based completely on built-in knowledge.

### Specifying Task Environment

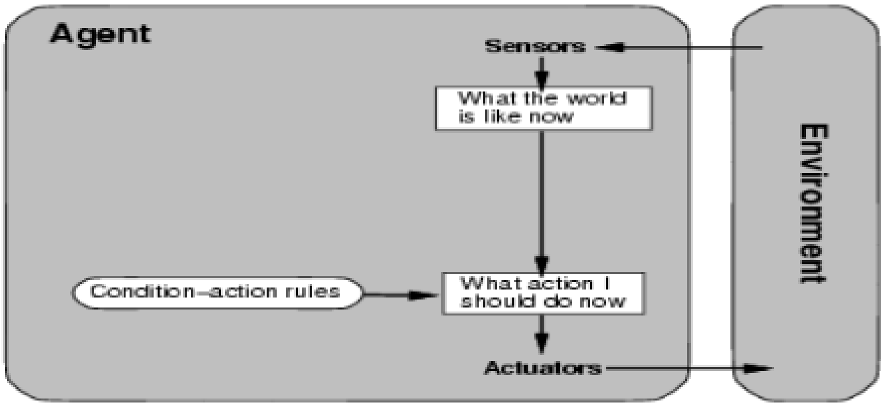
- `Task Environment` refers to the problems to which rational agents are the solutions.
- Specifying the task environment as fully as possible is always the first step in designing agent.
- **PEAS (Performance, Environment, Actuators, Sensors)**:
  - `Performance measure`: The performance measure that defines degrees of success of the agent.
  - `Environment`: Knowledge of its surrounding (What an agent already knows about the environment it works in).
  - `Actuators`: The actions that the agent can perform back to the environment.
  - `Sensors`: Everything that the agent has perceived so far concerning the current scenario in the environment.

### Agent Program Vs Agent Function

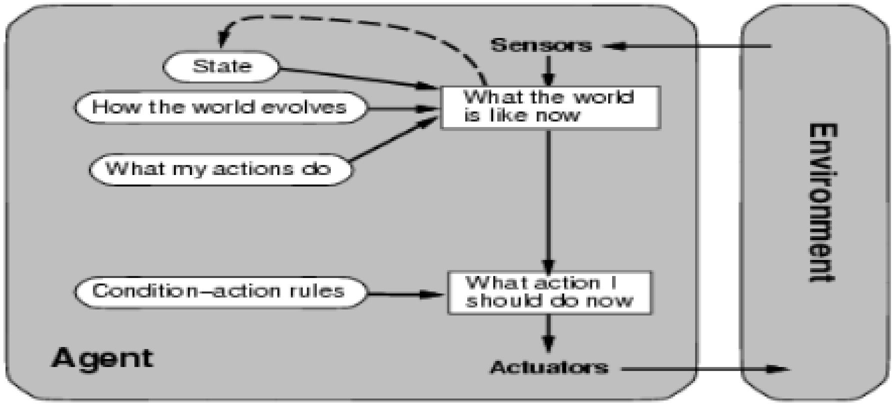
- Agent program takes the `current percept` as `input`, but Agent function takes the `entire percept history`.

### Classification of Agents

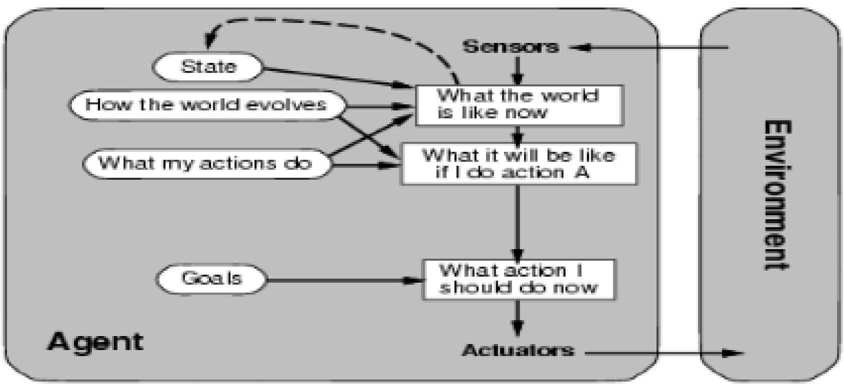
- Based on `memory of the agent`, and `the way the agent takes action`, agents are classified into 5 basic types:
  - **Simple reflex agents**:
    - Simplest type of Agent.
    - Only uses the current precepts.
    - Uses a set of `condition-action` rules ("if this is the percept then this is the best action").
    - They cannot make decisions on things that they cannot directly perceive, i.e. they have no model of the state of the world.
    - Very limited intelligence.



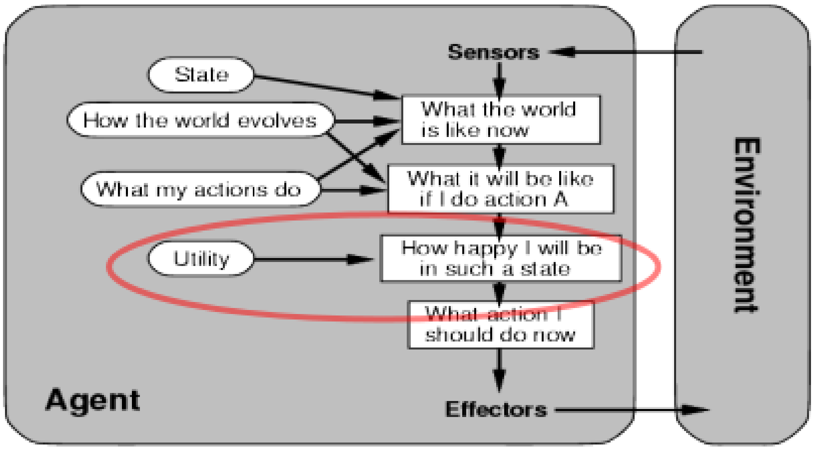

- **Model-based reflex agents**:
  - Complex type of Agent.
  - Maintain an internal model of the world, which is updated by precepts as they are received. In addition, they have built-in knowledge (i.e. prior knowledge) of how the world tends to evolve.
  - Since they live in the present only, they aren't capable of planning longer term goals.



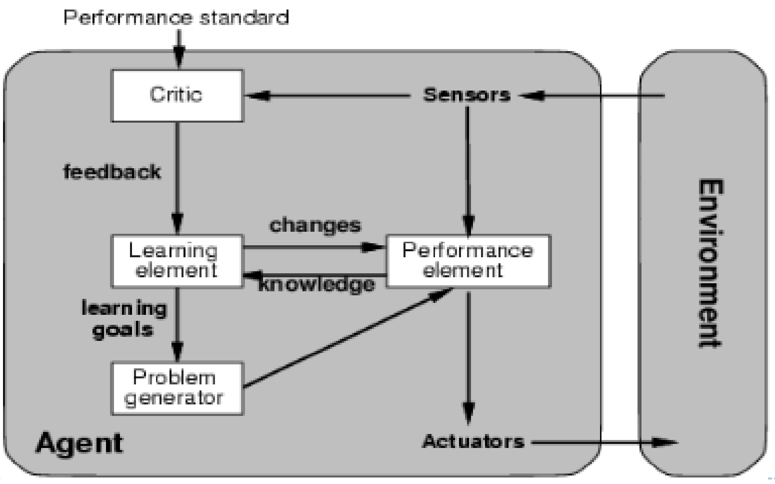

- **Goal-based agents**:
  - It keeps track of the world state as well as a set of goals it is trying to achieve, and chooses an action that will (eventually) lead to the achievement of its goals.




- **Utility-based agents**:
  - Utility based agents deal with this by assigning a utility to each state of the world.
  - This utility defines how "happy" the agent will be in such a state.
  - Unlke goal based agents that implicitly contain a utility function which is difficult to define more complex "desires", utility based agents explicitly state the utility function to make it easier to define the desired behaviour of utility based agents.




- Learning agent:
  - Preferred method for creating state of the art systems.
  - Components:
    - `Learning Element` :
      - Suggesting improvements to any part of the performance element.
      - The input to the learning element comes from the Critic.(on how the agent is doing and determines how the performance element should be modified to do better in the future).
    - `Performance Element` : Responsible for selecting external actions(it takes in percepts and decides on actions).
    - `Critic` :
      - Analyses incoming precepts and decides if the actions of the agent have been good or not.
      - To decide this it will use an external performance standard.
    - `Problem Generator` :
      - Responsible for suggesting actions that will result in new knowledge about the world being acquired.




### Environments

- An environment in artificial intelligence is the surrounding of the agent.

- **Types of Environments**:
  - Fully Obervable vs Partially Observable:
    - In a `fully observable` environment, The Agent is familiar with the complete state of the environment at a given time. There will be no portion of the environment that is hidden for the agent.
    - In a `partially observable environment` , The agent is not familiar with the complete environment at a given time.
  - Deterministic vs Stochastic:
    - `Deterministic` environments are the environments where the next state is observable at a given time. So there is no uncertainty in the environment.
    - The `Stochastic` environment is the opposite of a deterministic environment. The next state is totally unpredictable for the agent. So randomness exists in the environment.
  - Episodic vs Sequential:
    - `Episodic` is an environment where each state is independent of each other. The action on a state has nothing to do with the next state.
    - `Sequential` environment is an environment where the next state is dependent on the current action. So agent current action can change all of the future states of the environment.
  - Static vs Dynamic:
    - The `Static` environment is completely unchanged while an agent is precepting the environment.
    - `Dynamic` Environment could be changed while an agent is precepting the environment. So agents keep looking at the environment while taking action.
  - Discrete vs Continous:
    - `Discrete` Environment consists of a finite number of states and agents have a finite number of actions.
    - While in a `Continuous` environment, the environment can have an infinite number of states. So the possibilities of taking an action are also infinite.
  - Single Agent vs Multi Agent:
    - `Single agent` environment where an environment is explored by a single agent. All actions are performed by a single agent in the environment.
    - If two or more agents are taking actions in the environment, it is known as a `multi-agent` environment.
  - Known vs Unknown:
    - In a `known` environment, the output for all probable actions is given. Obviously, in case of `unknown` environment, for an agent to make a decision, it has to gain knowledge about how the environment works.

# Chapter 3: Problem Solving

### Problem Solving Agent

- A type of `goal-based` agent that decide what to do by finding sequences of actions that lead to the desirable states.
- Four general steps in problem solving:
  - `Goal formulation` : What are the successful world states.
  - `Problem formulation` : What actions and states to consider given the goal.
  - `Search` : Determine the possible sequence of actions that lead to the states of known values and then choosing the best sequence.
  - `Execute` : Give the solution perform the actions.

### Problem Definition

- A problem can be defined formally by five components:
  - The `initial state` that the agent starts in.
  - A description of the possible actions available to the agent at each state.
  - `Transition model` is a description of what each action does (Successor).
  - The `goal test` , which determines whether a given state is a goal state.
  - A `path cost` function that assigns a numeric cost to each path.
- `Solution` is a path from the initial state to a goal state.
- `Optimal solution` is the path that has the lowest path cost among all solutions.
- The process of removing detail from a representation is called `abstraction` .

### Types of Problems

- **Single-state problem**:
  - The environment is `Deterministic` and `fully observable` .
  - Out of the possible state space, agent knows exactly which state it will be in; solution is a sequence.
- **Sensor less problem(conformant problem)**:
  - The environment is `non-observable` .
  - It is also called `multi-state` problem.
  - Agent may have no idea where it is; solution is a sequence.
- **Contingency problem**:
  - The environment is `non-deterministic` and/or `partially observable` .
  - It is not possible to know the effect of the agent action.
  - Percepts provide new information about current state.
- **Exploration problem**:
  - The environment is partially observable.
  - It is also called unknown state space.

| Environment Type | Problem Type |
|---|---|
| Deterministic, fully-observable | Single-state problem |
| Non-observable, known state space | Sensor less/conformant problem |
| Nondeterministic and/or partially-observable | Contingency problem |
| Partially observable, unknown state space | Exploration problem |

### Searching

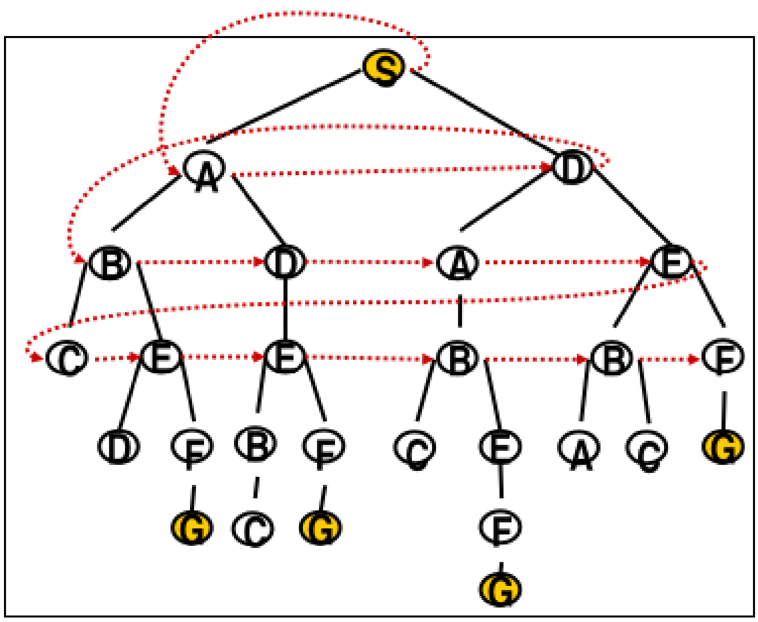
- The possible action sequences (solutions) starting at the initial state form a search tree with:
  - The `initial` state at the root.
  - The branches are `actions` applied on current state to create a new state and the `nodes` correspond to states having no successor.
- Search algorithms all share this basic structure, they vary primarily according to how they choose which state to expand next the so-called `search strategy` .
- A `search process` can be viewed as building a search tree over the state space.
- `Search tree` is a tree structure defined by initial state and a successor function.
- `Search(root) Node` is the root of the search tree representing initial state and without a parent.
- A `child node` is a node adjacent to the `parent node` obtained by applying an operator or rule.

### Search strategies

- A search strategy is defined by picking the order of node expansion.
- Strategies are evaluated along the following dimensions:
  - `Completeness` : does it always find a solution if one exists?
  - `Time complexity` : number of nodes generated.
  - `Space complexity` : maximum number of nodes in memory.
  - `Optimality` : does it always find a least-cost solution?
- Time and space complexity are measured in terms of:
  - `b` : maximum branching factor of the search tree.
  - `d` : depth of the least-cost solution.
  - `m` : maximum depth of the state space (may be ∞).

### Classification of search strategies

- Searching strategies can be classified in to two as `uninformed` and `informed` search strategies.
- **Uninformed search strategy(Blind search)**:
  - Use only the information available in the problem definition.
  - They have no information about the number of steps or the path cost from the current state to the goal.
  - They can distinguish the goal state from other states.
  - Basic terms:
    - `Leaf Node` : is a node without successors (or children).
    - `Depth (d)` : of a node is the number of actions required to reach it from the initial state.
    - `Frontier or Fringe Nodes` : are the collection of nodes that are waiting to be expanded.
    - `Path cost` : of a node is the total cost leading to this node.
  - Types:
    - `Breadth First Search (BFS) :`
      - Uses no prior information, nor knowledge.
      - It tracks all nodes because it does not know whether this node leads to a goal or not.
      - Keeps on trying until it gets solution.
      - All nodes are expanded from the root node.
      - That is it is a simple strategy in which:
        - the root node is expanded first,
        - then all the successors of the root node are expanded next,
        - then their successors, and so on.

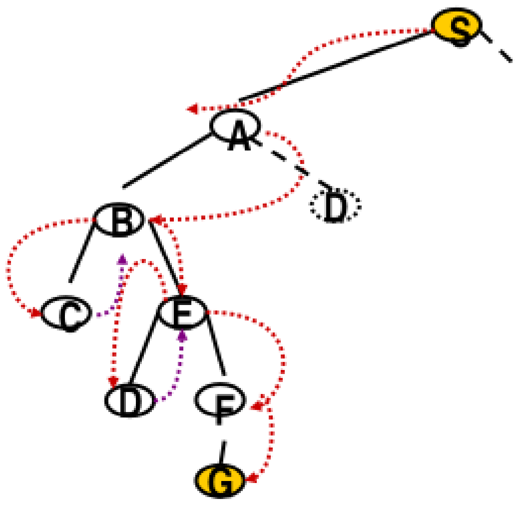



      - BFS algorthim uses FIFO policy.

- Properties:
  - Complete: Yes if $b$ is finite.
  - Time: $1 + b + b^2 + b^3 + .. + O(b^d)$
  - Space: $O(b^d)$
  - Optimal: Yes if $cost = k$ per step

- `Depth First Search (DFS)` :

  - Pick one of the children at every node visited, and work forward from that child.
  - Always expands the deepest node reached so far (and therefore searches one path to a leaf before allowing up any other path).
  - Thus, it finds the left most solution.




  - Properties:
    - Complete: Yes, if state space is finite.
    - Time: $O(b^m)$
    - Space: $O(b \times m)$
      - $b$ = max branching factor
      - $d$ = depth of the shallowest (least-soln) soln
      - $m$ = maximum depth of state space
    - Optimal: No

- `Backtracking search` :

  - A variant of depth first search (DFS).
  - Only one successor is generated at a time rather than all successors.
  - Each partially expanded node remembers which successor to generate next.
  - Memory requirement: only $O(m)$ vs. $O(bm)$

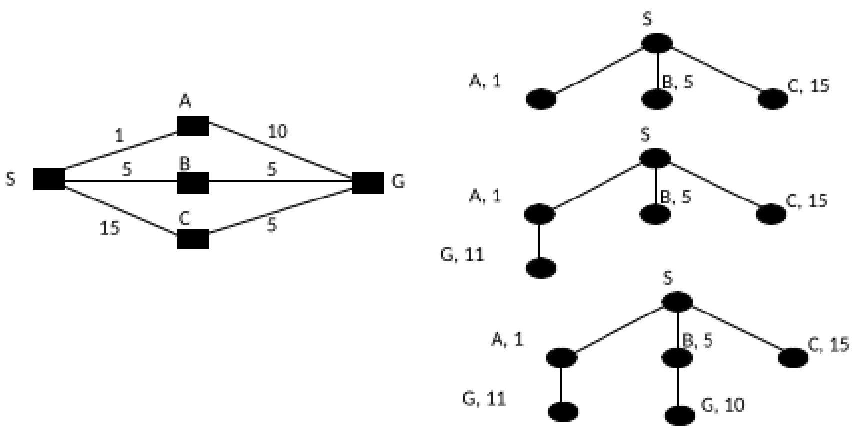
- `Depth Limited Strategy (Depth first search with cut off)` :

  - DFS with depth cutoff $k$ (maximal depth below which nodes are not expanded).
  - Three possible outcomes:
    - Solution
    - Failure (no solution)
    - Cutoff (no solution within cutoff)

- Solves the infinite-path problem.
- If $k < d$ then incompleteness results.
- If $k > d$ then not optimal.
- Properties:
  - Time: $O(b^k)$
  - Space: $O(b \times k)$
- DFS is a method of choice when there is a known (and reasonable) depth bound, and finding any solution is sufficient.

- `Iterative Deepening search` :
  - It tries all possible depth limits: first 0, then 1, 2, and so on.
  - Combines the benefits of depth-first (DFS) and breadth-first (BFS) search.
  - Suitable for the problem having a `large search space` and the `depth of the solution is not known` .
  - Properties:
    - Complete: Yes
    - Time: $O(b^d)$
    - Space: $O(b \times d)$
    - Optimal: Yes, if step cost = 1

- `Uniform cost search` :
  - Expand least-cost unexpanded node.
  - Equivalent to breadth-first if step costs all equal.




- `Bidirectional search` :
  - Bidirectional search is a graph search algorithm which find smallest path from source to goal vertex.
  - It runs two simultaneous search:
    - `Forward search` from source/initial vertex toward goal vertex.
    - `Backward search` from goal/target vertex toward source vertex.
  - We can consider bidirectional approach when:
    - Both initial and goal states are unique and completely defined.
    - The branching factor is exactly the same in both directions.

- Properties:
  - Complete: Yes,
  - Optimal: Yes
  - Time: $O(b^{d/2})$
  - Space: $O(b^{d/2})$

### Comparision of Uninformed search strategies

| Criterion | Breadth-First | Uniform-Cost | Depth-First | Depth-Limited | Iterative Deepening | Bidirectional (if applicable) |
|---|---|---|---|---|---|---|
| Complete? | Yes$^a$ | Yes$^{a,b}$ | No | No | Yes$^a$ | Yes$^{a,d}$ |
| Time | $O(b^d)$ | $O(b^{1+\lfloor C^*/\epsilon \rfloor})$ | $O(b^m)$ | $O(b^\ell)$ | $O(b^d)$ | $O(b^{d/2})$ |
| Space | $O(b^d)$ | $O(b^{1+\lfloor C^*/\epsilon \rfloor})$ | $O(bm)$ | $O(b\ell)$ | $O(bd)$ | $O(b^{d/2})$ |
| Optimal? | Yes$^c$ | Yes | No | No | Yes$^c$ | Yes$^{c,d}$ |

**Figure 3.21** Evaluation of tree-search strategies. $b$ is the branching factor; $d$ is the depth of the shallowest solution; $m$ is the maximum depth of the search tree; $l$ is the depth limit. Superscript caveats are as follows: $^a$ complete if $b$ is finite; $^b$ complete if step costs $\geq \epsilon$ for positive $\epsilon$; $^c$ optimal if step costs are all identical; $^d$ if both directions use breadth-first search.