Artificial Intelligence

Chapter 1 : Intelligent Agents - Introduction & problem solving

Chapter 2 : Search Methods

Chapter 3 : Propositional logic & knowledge representation. Semantics, Inference rules

Chapter 4 : Acting logically :- Planning - Partial order, scheduling, Types of planning

Chapter 5 : Searching techniques - First order logic - Forward reasoning - Backward reasoning.

Artificial Intelligence :-

AI Comprises of two words :-

(i) Artificial :- something made by human/non natural thing

(ii) Intelligence :- Ability to understand or think.

- Studies shows how to achieve intelligent behavior through computational means
- This makes AI a branch of Computer Science

John McCarthy, Coined the term in 1956 and defines it as "the science and engineering of making Intelligent machines, especially intelligent computer programs".

The definitions of AI are categorized into 4 approaches.

(i) systems that think like human :- Making m/cs to think as if machines are having sense in full and literal sense.

(ii) systems that think rationally :- Having reasn or understanding

(iii) systems that act like human :- Functions similar to human

(iv) systems that act rationally :-

In detail :-

(i) Acting humanly :- Eg, The Turing Test approach.

The Turing Test was proposed by Alan Turing in 1950. Called as the imitation Game

Defn :- "Whether m/c's can think like human"
ask questions of two entities - man & computer and receive answers from both.
If you can't tell which of the entities is human and which is a computer program, then you are fooled and this computer is considered to be intelligent.

For the computer to pass, it need to possess the following capabilities :

(i) Natural language processing
(ii) Knowledge representation :- To store what it knows or learns
(iii) Automated reasoning :- To use the stored info. to answer q, a

John McCarthy, Coined the term in 1956 and defines it as "the science and engineering of making Intelligent machines, especially intelligent computer programs".

The definitions of AI are categorized into 4 approaches

(i) systems that think like human :- Making m/cs to think as if machines are having sense in full and literal sense.

(ii) systems that think rationally :- Having reasoning or understanding

(iii) systems that act like human :- Functions similar to human

(iv) systems that act rationally :-

In detail :-

(i) Acting humanly :- Eg, The Turing Test approach.

The Turing Test was proposed by Alan Turing in 1950. Called as the Imitation Game

Defn :- "Whether m/c's can think like human"
ask questions of two entities - man & computer and receive answers from both.
If you can't tell which of the entities is human and which is a computer program, then you are fooled and this computer is considered to be intelligent.

For the computer to pass, it need to possess the following capabilities:

(i) Natural language processing
(ii) Knowledge representation :- To store what it knows or learns
(iii) Automated reasoning :- To use the stored info. to answer questions

iv) Machine Learning :- To adapt to new circumstances and to detect and extrapolate patterns

To pass the complete Turing Test, the computer will need :-

(ii) Computer vision → to perceive the objects

(iii) Robotics → to manipulate objects and move about

✓ Google AI has passed Turing Test

Computer (player A)	Human Respondent (player B)
Interrogator player C.	

Chatbots attempted Turing Test :-

1) ELIZA → NLP Computer program, which can communicate with human. ← DOCTOR 200 line program that made a user to understand that he/she is talking to machine

2) PARRY → simulate a person with most common Chronic mental disorder. } Didn't pass Turing

3) Eugene Goostman .. convinced 29 % of judges → 13 year old boy from Ukraine

Thinking Humanly : The Cognitive Modeling approach :

(i) through introspection :- Trying to capture our thoughts

(ii) through psychological experiments :- GPS → checked with reasoning steps by user

c) Thinking rationally :- This is Prescriptive
Eg) Right thinking.

Eg) Socrates is a man
all men are mortal
∴ Socrates is mortal ⟹ Logic.

d) Acting Rationally :- Doing the Right thing.

Eg) Computer Agents :-
- Operating under autonomous control
- Perceiving their environment
- Persisting over a prolonged time period
- Adapting to change.

Foundations of AI :

The various disciplines which contributed ideas & techniques are ;

(i) Philosophy :- (i) How does mental mind arise from physical brain?
(ii) Can formal rules be used to draw valid conclusions?
iii) Where does knowledge come from?
iv) How does knowledge lead to action?

Aristotle (384 - 322 B.C) was the first to formulate a precise set of laws governing the rational part of mind. ~~He developed~~ Philosophers (400 B.C) made AI possible, by considering the ideas that make us to feel like mind is some similar to box.

2) Mathematics : Formal representation & proof algorithms, computation & probability

Deals with (i) Formal rules to draw valid conclusions
(ii) What can be computed ?
(iii) How do you deal with Uncertain

3. Economics
   - How do we make decisions so as to maximize profit.
   - How do you do this when the payoff may be far in the future.
4. Neuroscience
   How do brains process info.
5. Psychology. How do humans and animals think and act
6. Computer Engineering :-
   For AI to succeed intelligence and artifacts are needed. AI programs tend to be large which requires good memory and CPU's.
7. Control theory and cybernetics :
   Self-controlling machines.
8. Linguistics : Language use fits in a model. Modern linguistics & AI are called as computational linguistics or Natural language processing.

History of AI :

Alan Turing articulated a complete vision of AI in 1950 by article "Computing Machinery and Intelligence". Then he introduced the Turing Test, Machine learning, Genetic algorithms, reinforcement algorithm.
"GPS was the computer program to implement "thinking humanly" approach.

LISP was invented by John McCarthy in 1958 while he was at MIT. In 1963 McCarthy started the AI lab at Stanford.

Dendral was the expert system project based on AI & it was released in 1960s. AI become an industry in 1980 till now.

1985 – 1995 : Neural N/w gained its popularity.

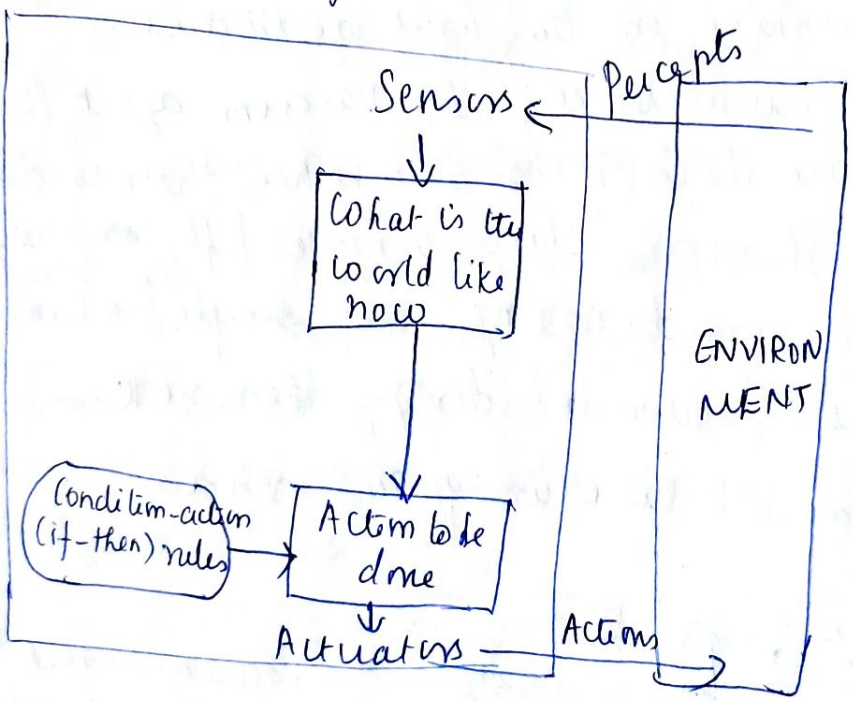
1988 – : Soft Computing.

In 2003 Human-level AI started emerging.

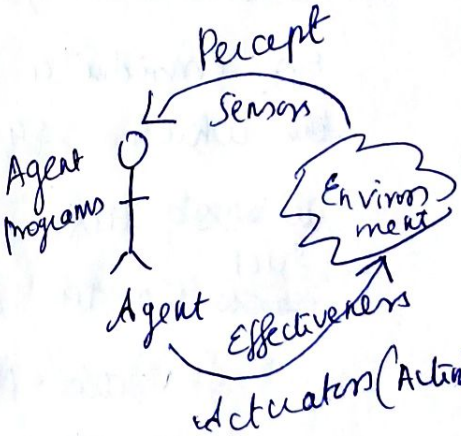
Due to the availability of Internet, high-speed processor and GPU's, ~~AI~~ is emerging with its charm in recent times.

Intelligent Agent:

An agent is anything that can be viewed as perceiving its environment through sensors and sensors acting upon that environment through actuators.



Goals of Agent
1. High performance
2. Optimized result
3. Rational actions



Agent → human, m/c, Robot, Program

Agents interact with environment through sensors & actuators

Agent → Percepts → Decisions → Actions

- In human agent: Sensors (eyes, ears), Actuators (hands, legs & mouth)
- Robotic agent: Sensors (cameras, infrared range finders), Actuators (motors)
- Software Agent: Sensors (keystrokes, file contents, network packets), Actuators (displaying on screen, writing files, sending network packets)

PERCEPT:

Percept refers to the agent's perceptual inputs at any given instant.

Percept Sequence:

An agent's percept sequence is the complete history of everything the agent has ever perceived.

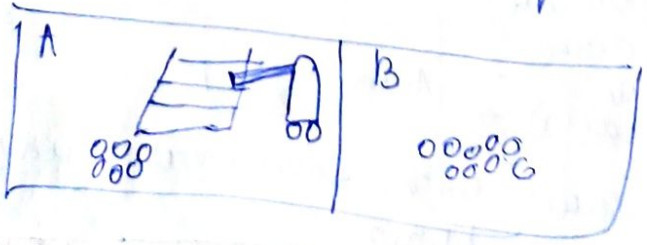
Agent function:- It is a map from the percept sequence to an action.

Agent program: The agent function for an artificial agent will be implemented by an agent program. The agent program is a concrete implementation, running in the agent architecture.

Eg) Consider a vacuum cleaner. The vacuum agent perceives which square it is in and whether there is dirt in the square. It can choose to move left, move right, suck up the dirt or do nothing. One simple function is:

If the agent function is dirty, then suck.

Otherwise move to the other square



Vacuum cleaner world with two locations.

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

Agent function for vacuum cleaner.

Concept of Rationality:

An agent should act as a Rational agent. A rational agent is one that does the right thing inorder to make the environment successful.

Performance measures:

What is rational at any given time depends on 4 things:

Defn of Rational agent

(i) The performance measure that defines the criterion of success
(ii) The agent's prior knowledge of the environment
(iii) The actions that the agent can perform
iv) The agent's percept sequence to date.

Omniscience, Learning and Autonomy:-

Omniscient agent :- knows the actual outcome of its actions & can act accordingly; but impossible in reality.

A rational agent not only gather info., but also learn as much as possible. Successful agents split the task of computing agent function into 3 different periods:

(i) when the agent is being designed.
ii) when it is deliberating on its next action
iii) As it learns from experience.

Nature of the Environment

specifying the task environment : (PEAS)

Performance, Environment, Actuators, Sensors

Taxi Task environment

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, GPS, Speedometer, odometer engine sensors keyboards, Accelerometer

Properties of task Environment.

(i) Fully observable / partially observable.

If an agent's sensors give it complete access to the complete environment at each point in time.

An environment may be partially observable because of noisy / inaccurate sensors or because parts of states are simply missing from the sensor data

ii) Deterministic Vs non-deterministic (or) stochastic :

Deterministic: If the next state is completely determined by the current state

iii) Episodic Vs non-episodic (or) sequential :

Episodic → Agent's experience is divided into atomic episode. Each episode consists of its own Percepts and actions and it does not depend on previous episodes.

In sequential, the current decision could affect all future decisions. Eg) Chess / Taxi driving

iv) Static vs dynamic:

Dynamic: If the environment is changing for agent's action

Semi-dynamic: If the environment doesn't change & change after some time.

Eg) Taxi driving is dynamic
Chess is semi dynamic
Crossword puzzles are static

v) Discrete Vs Continuous:

Discrete: The environment has limited no. of distinct, clearly defined percepts & actions

Eg) Chess

Continuously: The environment changes continuously with range of values.

Eg) Taxi driving.

Structure of agent:

An intelligent agent is a combination of Agent Program & Architecture.

Intelligent agent = Agent Program + Architecture

Agent Program → function that implements the agent mapping from Percepts to actions.

Architecture → Computing device used to run the agent program.

Types of Intelligent agents:

1. simple reflex agent
2. Model based reflex agents
3. Goal based reflex agents
4. Utility based reflex agents
5. Learning agents

1) simple reflex agents:

- It responds directly to percepts

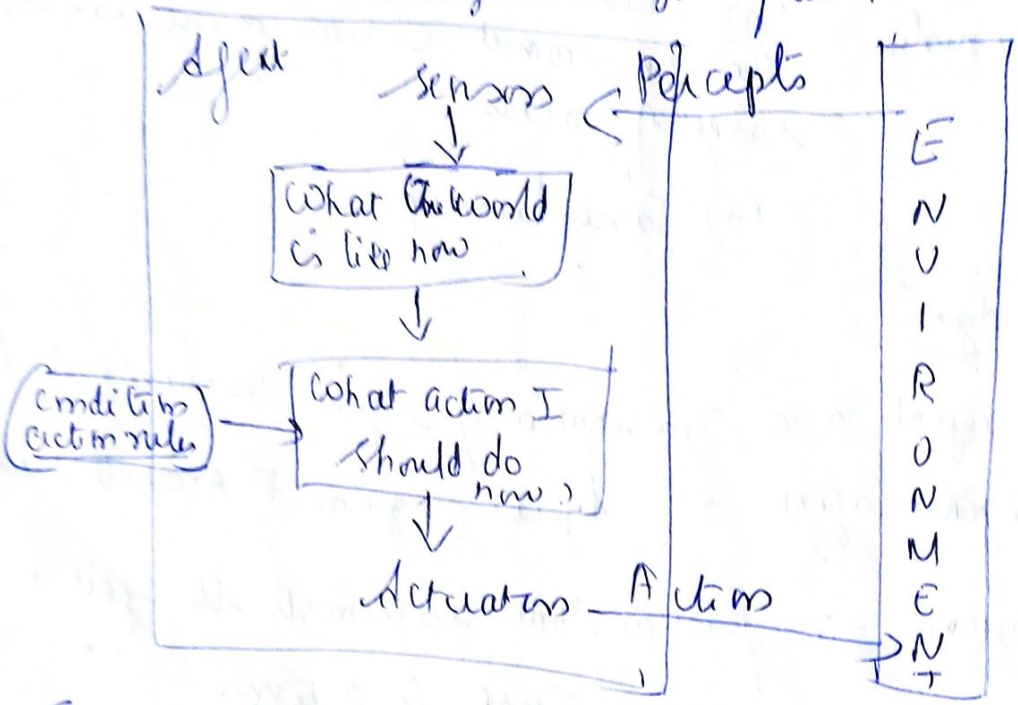
i.e, these agent select actions on the basis of current Percept, ignoring the rest of the percept history.

An agent describes about how the Condition action rules allow an agent to make the connection from percept to action.

Condition action rule: if Condition then action

Rectangle → current internal state of the agents decision process

Oval → background info. of the process.



Eg) Agent intelligent program:

function SIMPLE-REFLEX-AGENT (Percept) returns an action

static: rules, a set of condition-action rules

state ← INTERPRET-INPUT (Percept)

rule ← RULE-MATCH (state, rules)

action ← RULE-ACTION [rule]

return action

INTERPRET-INPUT → function generates an abstracted description of the current state from the percept

RULE-MATCH → function returns the first rule in the set of rules that matches the given state description

RULE-ACTION → the selected rule is executed as the action of the given percept.

The correct decision is made on the basis of current percept. i.e, if the environment is fully observable.

Eg) Medical diagnosis system

If the patient has reddish brown spots then start the treatment for measles.

2. Model based agents (Agents that keep track of the world)

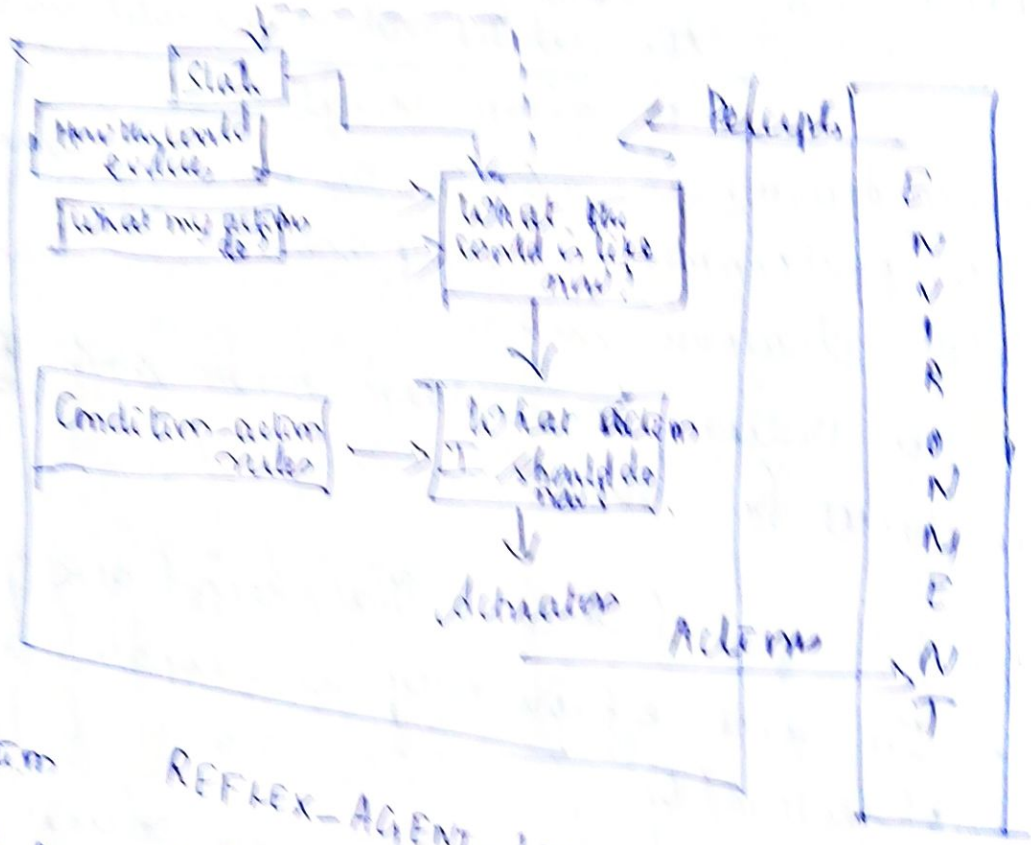
- The most effective way is to handle partial observability
- The agent which combines the current percept with the old internal state to generate updated description of the current state.
- The current percept is combined with the old internal state and it derives a new current state which is then updated in the state description

Updations requires 2 kinds of knowledge:

(i) how the world evolves independently of the agent

(ii) Information regarding how agents affect the world

The above knowledge is implemented in simple Boolean circuits or in complete scientific theories. Called as model of the world. An agent that uses such a model is called as Model-based agent.



function REFLEX-AGENT-WITH-STATE (Percept) returns an action
state : state, a description of the current world state
rules, a set of condition-action rules
action, the most recent action, initially none
state ← UPDATE-STATE (state, action, Percept)
rule → RULE-MATCH (state, rules)
action → RULE-ACTION [rule]
return action

UPDATE-STATE - Responsible for creating new internal state description by combining Percept and

3. Goal based agents :- (planning)

- Expansion of Model based reflex agents
- Desirable situations (Goal)
- Searching & Planning

} eg) supervised

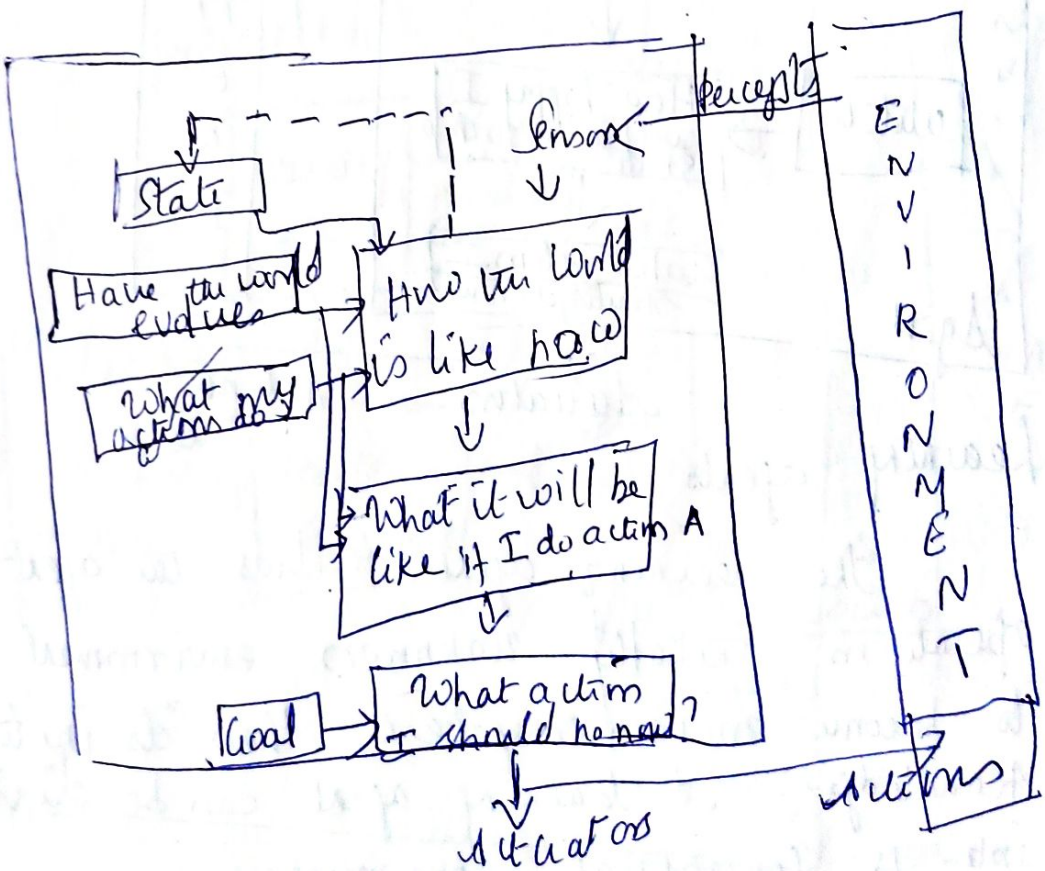
① The knowledge of the current state environment is not always sufficient to decide for an agent to what to do.

② The agent needs to know its goal.

③ Expansion of model based agents with "goal information".

④ They choose an action, so that they can achieve the goal.

⑤ These agents have to consider a long sequence of possible actions before deciding whether the goal is achieved or not. Such considerations of different scenario are called Searching and planning, which makes an agent proactive.



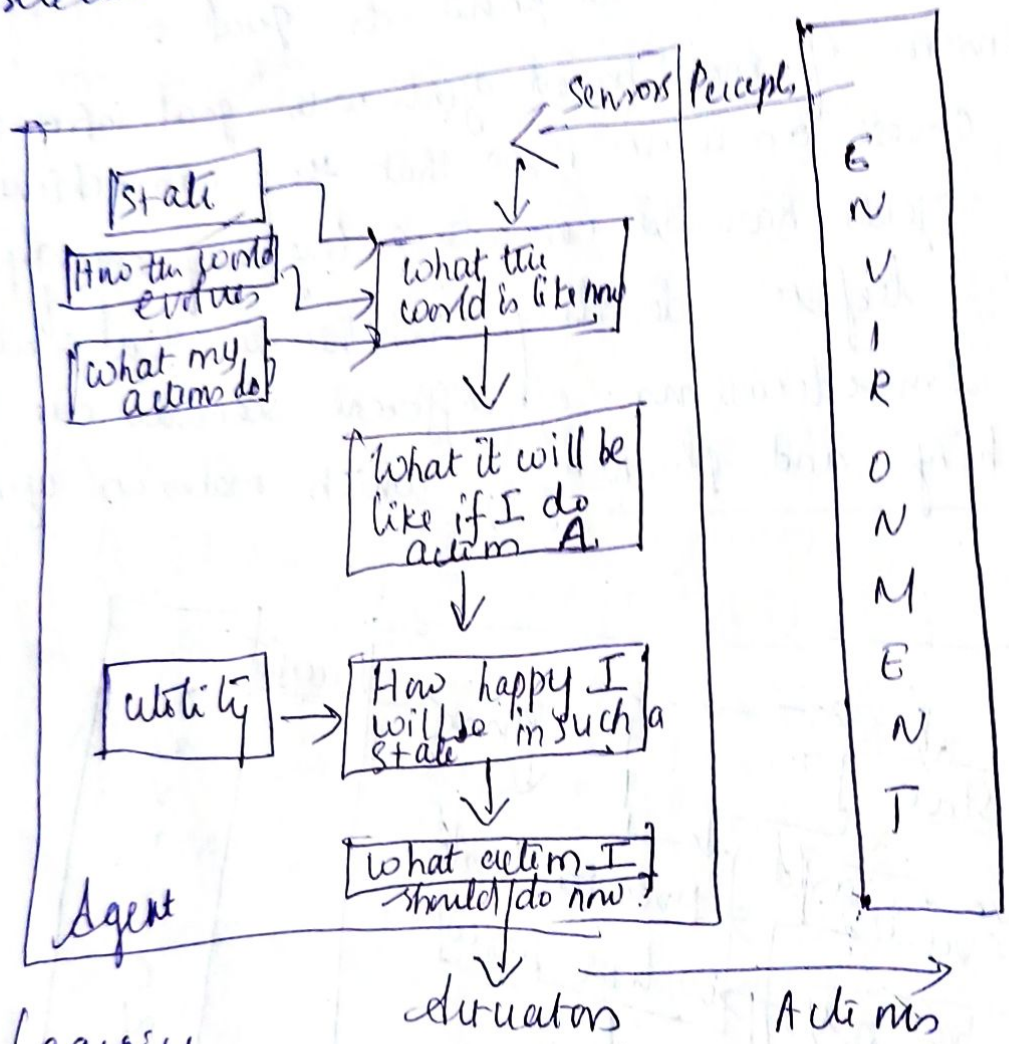
Can change goal without need to reprogram rules.

GPS

6)

4) Utility based agents.

An agent generates a goal state with high quality behaviour. If more than one sequence reaches the goal state then the sequence with more reliable, safer and quicker and cheaper than others to be selected.



5. Learning agents:

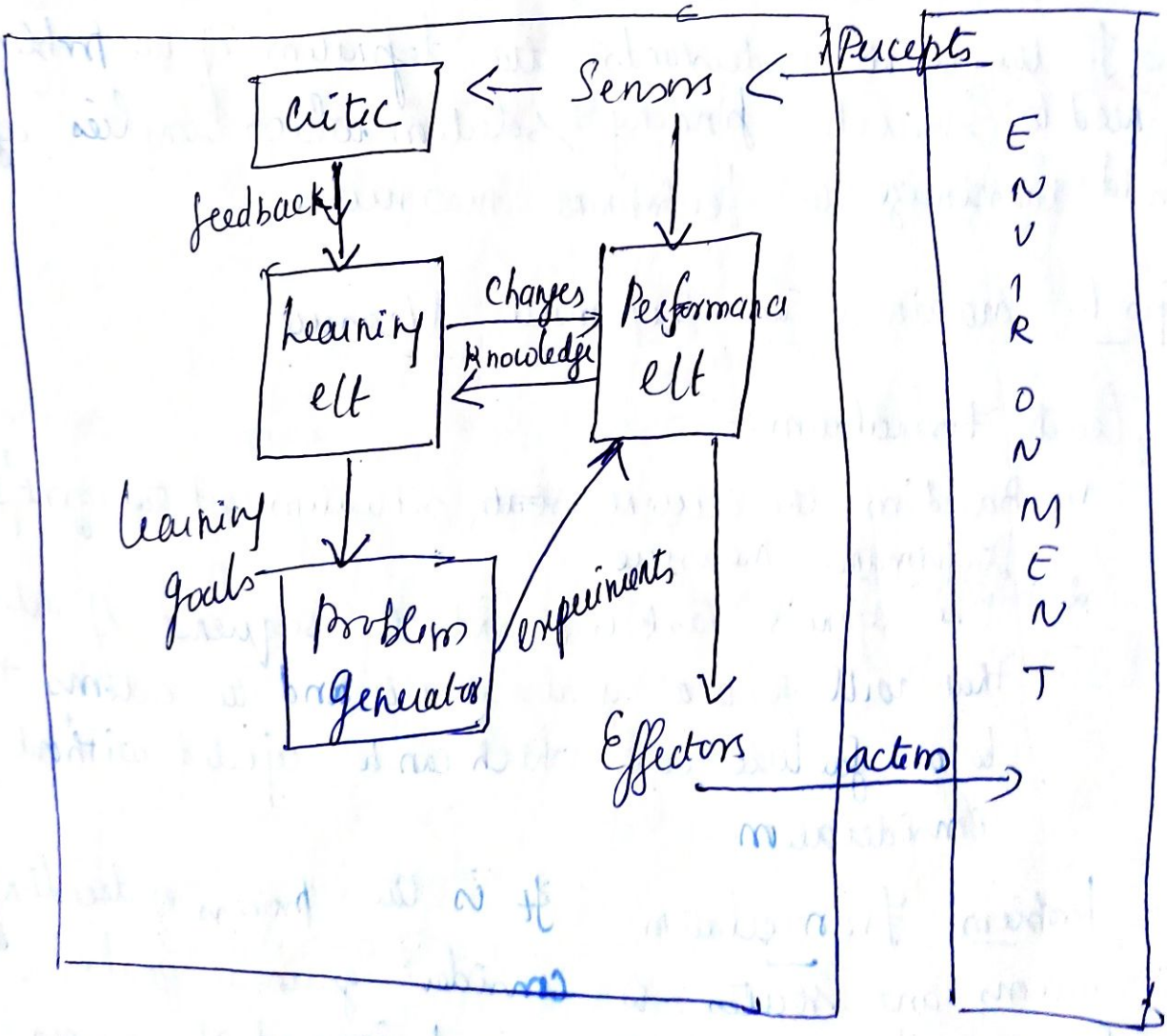
The learning agent allows the agent to operate in initially unknown environment and to become more competent than its initial knowledge. A learning agent can be divided into 4 conceptual components.

① Learning elt: This is responsible for making improvements. It uses the feedback from the critic on how the agent is doing and determines how the performance elt should be modified to do better in future.

2) Performance element: Responsible for selecting external actions and it is equivalent to agent. It takes in percepts and decide on actions

③ Critic :- It tells the learning element how well the agent is doing with respect to a fixed performance standard

4. Problem generator :- It is responsible for suggesting actions that will lead to new and informative experiences



PROBLEM SOLVING AGENTS.

An important aspect of intelligence is goal-based problem solving solution of many problems can be described by finding a sequence of actions that lead to a desirable goal.

Each action changes the state and the aim is to find sequence of actions and states that lead from the initial (start) state to a final (goal) state.

Problem formulation:

Problem solving is one type of goal based agent, where the sequences of actions, the agent should select one from these actions which lead to desirable states.

If the agent understands the definition of the problem, the we need to search finding solution which implies agent should maximize the performance measures.

Steps to maximize the performance Measure:

1. Goal Formulation :-
   1. Based on the current ~~state~~ situation and the agent's performance measure
   2. The agent's task is to find the sequence of actions that will go to a goal state and the actions that are to a failure case which can be rejected without further consideration.
2. Problem Formulation : It is the process of deciding what actions and states to consider given a goal.

Search :- An agent with several immediate options of

unknown value can examine different possible sequences of actions that leads to the states of known value and then chooses the best sequence.

The process of looking for sequences actions from the current state to reach the goal state is called Search.

4. Solution :- The search algm takes a problem as input & returns a solution in the form of action sequence.

5. Execution :- Once a solution is found, the execution phase consists of carrying out the recommended action.

```
function SIMPLE-PROBLEM-SOLVING-AGENT (percept) returns an action
inputs : percept, a percept
static : 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 do
        goal ← FORMULATE-GOAL (state)
        problem ← FORMULATE-PROBLEM (state, goal)
        seq ← SEARCH (problem)
    action ← FIRST(seq)
    seq ← REST (seq)
    return action
```

State

Observable

Discrete

Deterministic

An agent carries out its plan with eye closed. It is called an open loop system.

Well defined problems & solutions.

1) The initial state

2) A successor function returns possible actions available to the agent.

3) Goal test.

Determines whether the given state is goal state

Eg Goal is to reach a state named as "checkmate".

4) Path cost:

Assigns numeric cost to each action.

a) Step cost: Taking action a to go from state x to state y

b) A solution to the problem is the path from the initial state to a goal state

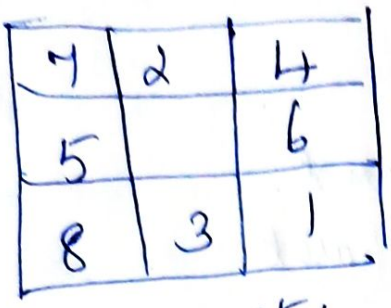
c) An optimal solution has the lowest path cost among all solutions

Toy problems
Eg. problems

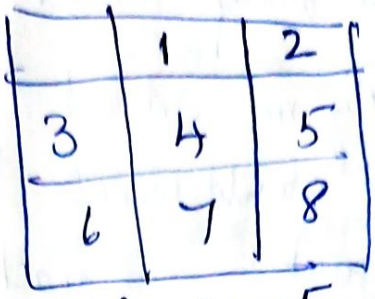
The problem solving approach has been applied to a variety of task environments
① Toy problems
② Real world problems

8 puzzle problem

8 puzzle - 3×3 board



Start state



Goal state

with 8 numbered tiles & a blank space.

States: Location of each of the eight tiles & a blank in one of nine squares

Initial state :- any state can be designated as the initial state.

Successor function : A legal ~~actions~~ states that result from trying the four actions (Left, Right, Up, down)

Goal state :- This checks whether the state matches the goal configuration as shown in Goal state diagram.

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

8 puzzle belong to the family of sliding-block puzzles. These are called as NP-complete

8 puzzle has $9!/2 = 181,440$ reachable states & can be easily solved

15 puzzle (4 * 4 board) has 1.3 trillion states. It can be solved in few milliseconds by the best search algms

24 puzzle (5 * 5 board) ~~can be solved~~ is difficult to solve.

2)

Eg) Romanian Holiday search problem

Eg) Route finding problem

Currently in Arad.

Flight should move to Bucharest

Formulate goal: To reach Bucharest

States: various state, sequence of cities: Arad, Sibiu, Fagaras, Buc

Initial state: Arad

Successor function: drive between cities

Goal Test: check whether Bucharest is reache

Path cost: Min. cost.

3) Vacuum World.

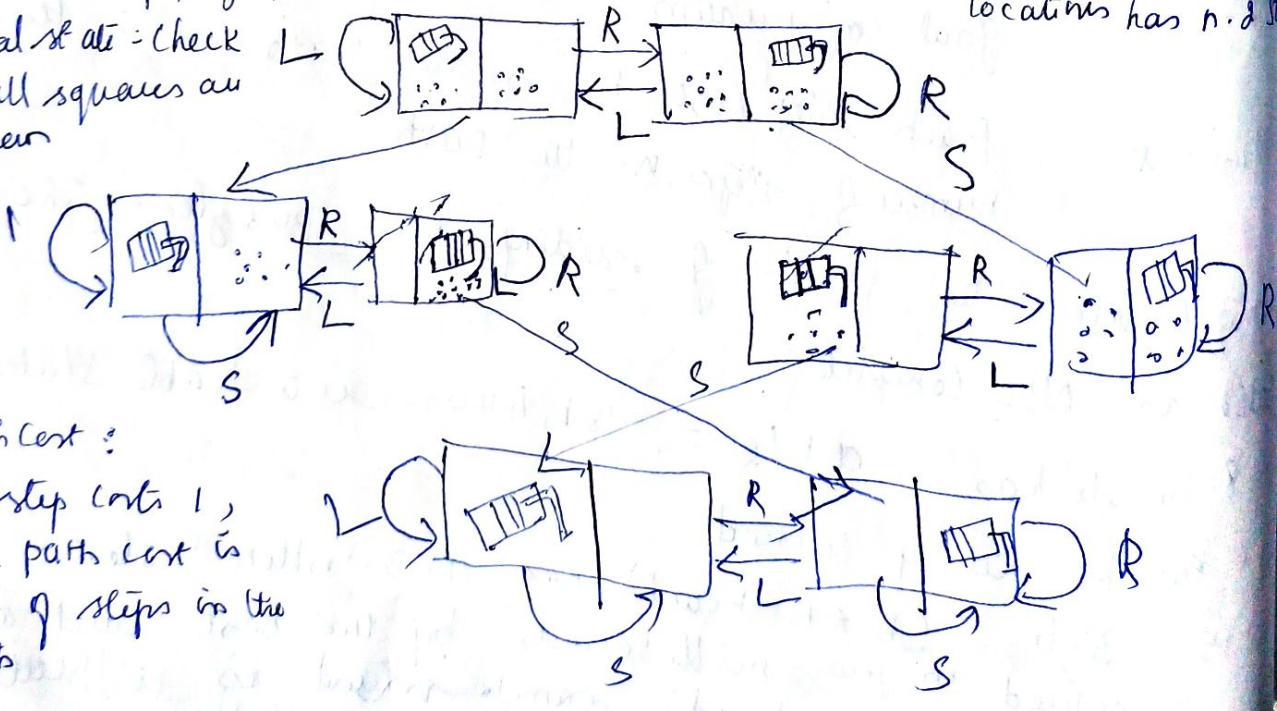
(i) There are $2 * 2 = 8$ possibl

A larger environment with n locations has n.2

(1) Initial State: Any state

ii) Actions: Left, Right, Suck

iii) Goal state: Check if all squares are clean



iv) Path cost:

Each step costs 1, so the path cost is the no. of steps in the path

4) 8 queens problem.

Place the queens on the board such that they do not attack with each other.

States : Any arrangement of 0 to 8 queens on a board
Initial state : No queen on board
Successor function : Add a queen to any empty square
Goal test : 8 queens on the board, none attacked

5 Real-world problems.

1) Route-finding problem. → computer n/w, military operations planning & airline travel planning systems

Air-line planning.

- Initial state :
- successor function :-
- Goal test : Are we at destination by some pre specified time?
- Path cost : monetary cost, waiting time, flight time, etc.

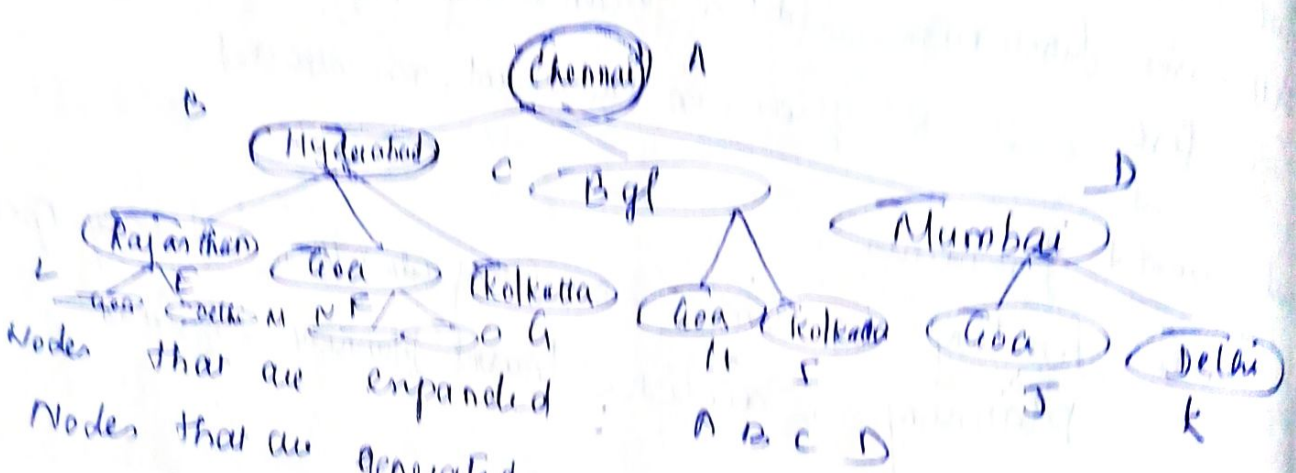
2) Travelling Sales person problem.
NP-hard.

Robot navigation
Automatic assembly navigation
Internet Searching

Searching for solutions

Search Tree: A search tree is generated by the
(i) Initial state
(ii) Successor function] ⟹ State space.

Eg) The full is a search tree for finding a route from Chennai to Delhi



Nodes that are expanded : A B C D

Nodes that are generated but not expanded : E F G H I J K

Nodes that have not yet been Generated : L M N O

Initial state: Root of the search tree

The first step is to test whether this is a goal state.
The current state is expanded by applying the successor function to the current state ⟹ new state.
The choice of which state to expand is determined by the search strategy.
There are infinite no of paths in the state space / so the search tree has an infinite no of nodes.

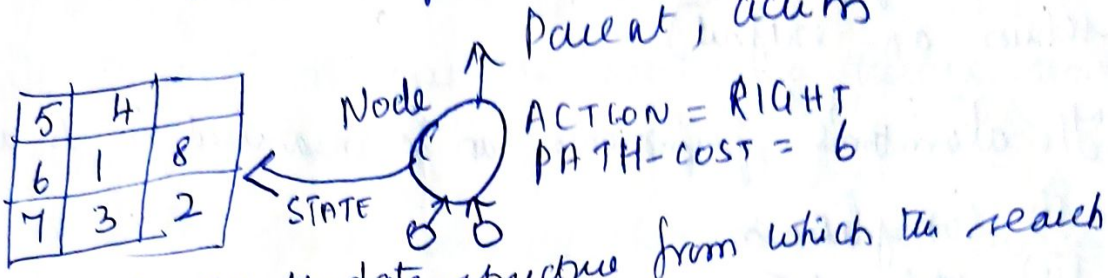
A Node is a data structure with 5 components.
(i) STATE : A state in the state space
(ii) PARENT NODE : the node in the search tree that generated this node.

iii) ACTION :- the action that was applied to the parent to generate the node.

iv) PATH-COST : the cost from parent node denoted by g(n)

v) DEPTH : the no. of steps along the path from the initial state.

Eg)



Nodes are the data structure from which the search tree is generated

Fringe : Fringe is a collection of nodes that have been generated but not yet been expanded. Each element of the fringe is a leaf node; ie, a node with no successors in the tree. The collection of these nodes are implemented as a queue

MAKE-QUEUE(element ...) ⟹ creates a queue with the given elts

EMPTY?(Queue) ⟹ returns true only if there are no more elts in the queue

FIRST(queue) ⟹ returns FIRST(queue) and removes it from the queue

INSERT(element, queue) ⟹ inserts an element into the queue and returns the resulting queue

INSERT-ALL(elements, queue) ⟹ inserts a set of elements into the queue and returns the resulting queue.

Measuring problem solving Performance

The o/p of problem solving algorithms is either failure or a solution. (Some algorithms might struck in an infinite loop and never return an output)

The algorithm's performance can be measured in 4 ways:

(i) Completeness
(ii) Optimality
(iii) Time Complexity
(iv) Space Complexity

Uninformed Search Strategies

Strategies that know whether one non goal state is "more promising" than another are called Informed Search or heuristic Search strategies.

(i) Breadth-first Search
(ii) Depth-first Search
(iii) Uniform-cost Search
(iv) Depth-limited Search
(v) Iterative deepening Search

(i) Breadth-first Search (BFS)

- Searches breadth-wise
- Traverses the tree where each node is a state which may be a potential candidate for solution

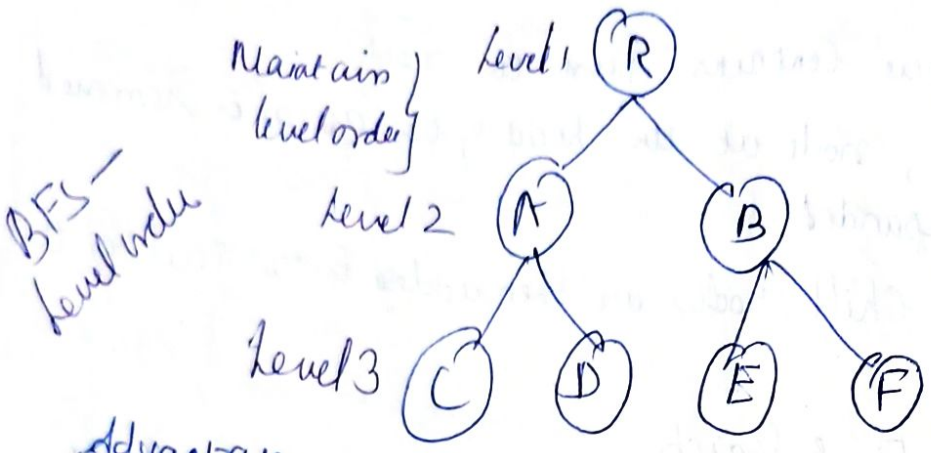
- It expands nodes from the root of the Tree and then generates one level of the tree at a time until a solution is found.
- Initially the queue contains just the root.
  In each iteration, node at the head of the queue is removed and then expanded.
- The generated child nodes are then added to the tail of the queue.

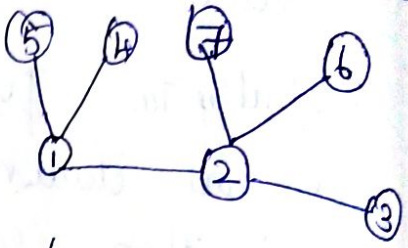
Algorithm: Breadth-First Search

1. Create a variable called NODE-LIST and set it to the initial state
2. Loop until a goal state is found or NODE-LIST is empty
   a) Remove the first element E, from the NODE-LIST
      If NODE-LIST was empty then quit.
   b) For each way that each rule can match the state described in E do:
      (i) Apply the rule to generate a new state
      (ii) If the new state is the goal state, quit and return this state.
      (iii) Otherwise add this state to the end of NODE-LIST.

Note:
BFS always finds a shortest path to a goal.
Since each node is generated in constant time, the amount of time used by Breadth first search is proportional to the no. of nodes generated, which is a

Since the no. of nodes at level d is b^d, the total no. of nodes generated in the worst case is $b + b^2 + b^3 + \ldots + b^d$ ie, $O(b^d)$





BFS: 1-2-4-5-6-3

From ① choose adj vertices. It can be in any order.

Advantages:

1. BFS will never stop exploring unwanted path
2. If there is a solution BFS will definitely find it
3. If there is more than one solution, BFS can find the minimal one that requires less no. of steps

Drawbacks:

1. Memory requirement: Since each level must be saved, the amount of memory is proportional to the no. of nodes stored.
   ∴ The space complexity of BFS is $O(b^d)$
2. If the solution is farther away from the root, BFS will consume lot of time.

2) Uniform Cost search (Find goal node with lowest cumulative cost)

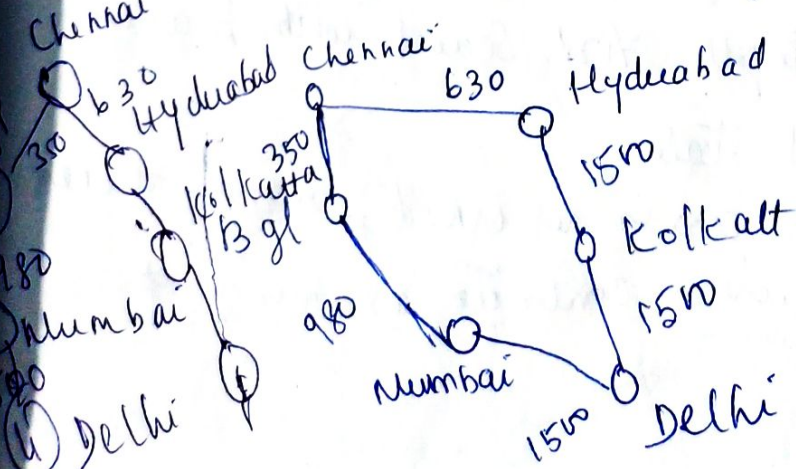
i) If all the edges in the search graph do not have the same cost, then BFS generalizes to uniform-cost search

ii) Instead of expanding nodes in the order of their depth from the root, uniform cost-search expands nodes in the order of their cost from the root.

iii) At each step, the next step n to be expanded is one whose cost g(n) is lowest where g(n) is the sum of the edge costs from the root to node n. The nodes are stored in a priority queue. This algorithm is also known as Dijkstra's single-source shortest algm.

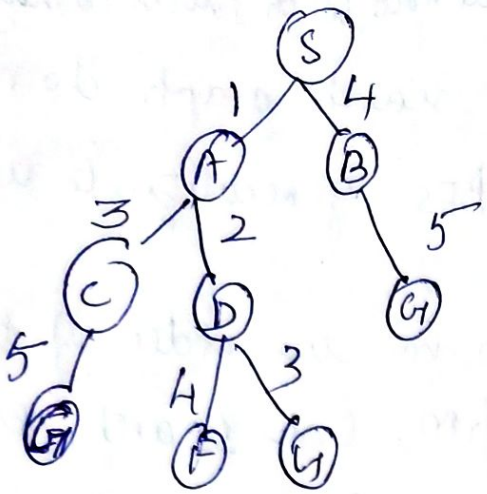
iv) Whenever a node is chosen for expanding by uniform cost search, a lowest-cost path to that node has been found.

v) The worst case time complexity of uniform cost-search is $O(b^{c/m})$, where c is the cost of an optimal solution and m is the min-edge cost.



Chennai - Bgl - mumbai - Delhi - 2830

Chennai - Hyderabad - Kolkatta - Delhi - 3630



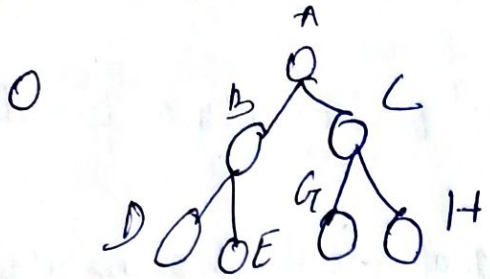
S – A — D – G

Depth first Search (DFS)

DFS Searches deeper into problem space. BFS always ~~last in~~ first-out stack for keeping the Unexpanded

Algorithm :- Depth first Search

1. If the initial state is a goal state, quit return success.
2. Otherwise loop until success or failure is signaled.
   (a) Generate a state say E and let it the successor of the initial state. If there is no successor, signal fa
   (b) Call Depth-First Search with E as the initial state
   c) If success is returned, signal suc Otherwise continue in this loop



Advantages :-

1) Memory requirement is linear with respect to Search graph, where as in BFS we require more space.

2) Time Complexity $O(b^d)$, since it generates the same set of nodes as BFS, but in different order.

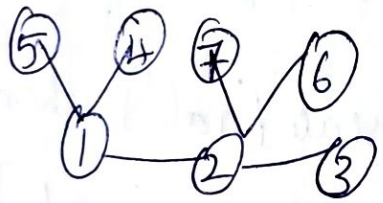
3) If DFS search finds solution without exploring much in a path, then time & space it takes will be less

Drawback.

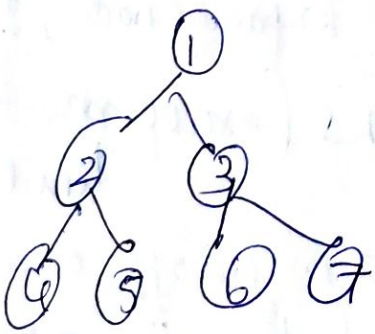
① Not guaranteed to find the Solution.

② There is no guarantee to find a minimal solution, if more than one solution exists.

Eg)



DFS: 1, 2, 3, 6, 7, 4, 5



1, 2, 4, 5, 3, 6, 7

Pre-order traversal.

4. Depth limited search (III to

DFS will not find a goal if it searches down a path that has infinite length. So in general, DFS is not guaranteed to find a solution. So it is not complete.

This problem is eliminated by limiting the depth of the search to some value l. This introduces another way of preventing DFS from finding the goal. The goal is deeper than l will not be found.

function Depth-Limited-Search (problem, limit) return solution

return Recursive-DLS (Make-Node (Initial-State [Prob problem, limit)

function Recursive-DLS (node, problem, limit) return solution / fail / cutoff

cut-off-occured ? false

IF Goal-Test (problem, state [node]) then

return Solution (node)

else if Depth [node] = limit then return cutoff

else for each successor in Expand (node, problem

result Recursive-DLS (successor, problem, limit)

if result = cutoff then cutoff_occured ?

else if result not = failure then return result

if cutoff_occured ? then return cutoff else return

Depth-limited search can be terminated with 2 conditions of failure:

(i) Standard failure value: It indicates that problem does not have any solution.

(ii) Cutoff failure value: It defines no solution for the problem within a given depth limit.

Advantages

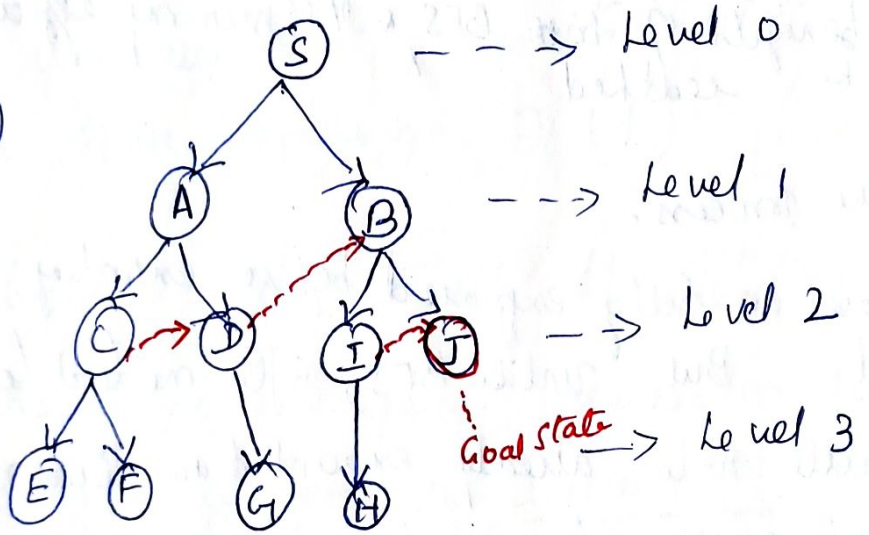
Memory efficient

Drawbacks

(i) incompleteness

(ii) If the problem has more than one solution, it may not be optimal.

Eg)



Limit - Level 2 ⇒ when level J is reached, Search terminates.

Time complexity = $O(b^l)$.

Space complexity = $O(b * l)$.

5. Iterative Deepening Search

If a depth-limited depth first search limited depth l doesn't find the goal, try it again with $l+1$. Continue this until goal is found.

Make depth-limited depth first search complete by repeatedly applying it greater values for the depth limit l.

It is a combination of BFS & DFS.

Advantage:

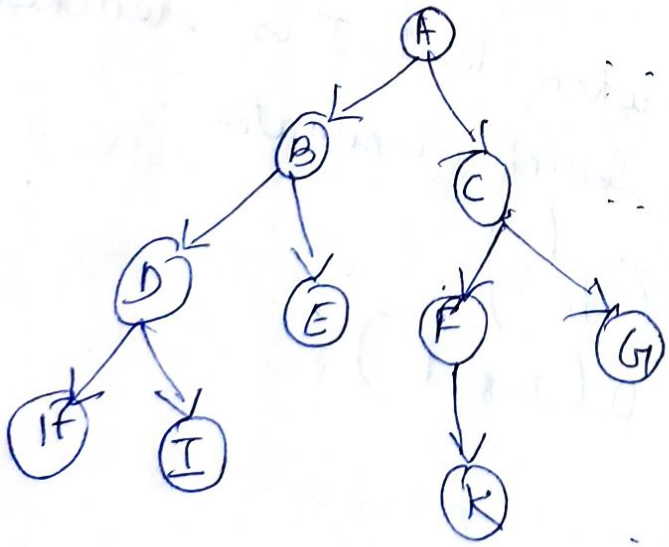
(fast search)

Incorporate benefits of both BFS & DFS (memory efficiency) Goal can be reached

Drawback:

Repeats the process.

Like BFS, a level is fully explored before extending the search to the next level. But unlike BFS, after one level is fully explored, all nodes already expanded are thrown away the search starts with a clear memory.



→ Level 0

→ Level 1

→ Level 2

→ Level 3

Assume K → Goal state

1st Iteration → A : not goal

2nd Iteration → A, B, C : not goal

3rd Iteration → A, B, D, E, C, F, G : not goal

4th Iteration → A, B, D, H, I, E, C, F, K, G → Goal state is reached

The algm is complete if the branching factor is finite.

Time Complexity:

branching factor : b

depth : d

worst case time complexity : $O(b^d)$

space complexity : $O(bd)$.

6. Bidirectional Search Algorithm

(i) As the name implies, searches in two directions at the same time one forward from the initial state and the other backward from the goal.

(ii) This is done by expanding tree with branching factor b and the distance from start to goal d.

(iii) The search stop when ~~two~~ searches from both directions meet in the middle.

iv) Bidirectional search is a brute-force search algm, that requires an explicit goal state instead of simply a test for a goal condition.

v) once the search is over, the path from the initial state is concatinated with the inverse of the path from the goal state to form the complete solution.

vi) Bidirectional search still guarantees optimal solution.

Time complexity : $O(b^{d/2})$ since each search need only to proceed to half the solution paths.

Space complexity : $O(b^{d/2})$

Advantages

(i) Speed. Sum of time taken by 2 searches (forward and backward) is much less than $O(b^d)$

(ii) It requires less memory.

Drawbacks

(i) Implementation of bidirectional search is difficult because additional logic must be included to decide which search tree to extend at each step.

(ii) Goal state need to be known in advance.

(iii) The algm must be too efficient to find the intersection of two search trees.

iv) It is not always possible to search backwards through possible states

Summary of algorithms

Criterion	BFS	DFS	Depth-limited	Iterative deepening	Bidirectional Search
Complete?	YES	No	YES, if $l \geq d$	YES	YES
Time	b^{d+1}	b^m	b^l	b^d	$b^{d/2}$
Space	b^{d+1}	b^m	b^l	b^d	$b^{d/2}$
Optimal?	YES	No	No	YES	YES

Searching with Partial information

If the knowledge of states / actions is incomplete, it leads to 3 distinct problems:

(i) Sensorless problem:- If no sensors, it could be in one of several possible initial states, and each action might lead to one of several possible successor states.

ii) Contingency problem. If the environment is partially observable or if actions are uncertain (adversarial) then the agent's percepts provide new information after each action. Each possible percept defines a contingency that must be planned for.

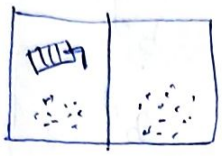
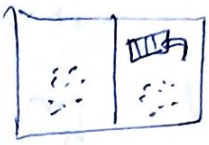
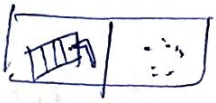
A problem is called adversarial if the uncertainty is caused by actions of another agent.

iii) Exploration problems:

When the states and actions of the environment are unknown, the agent must act to discover them. Exploration problems can be viewed as an extreme case of contingency problems.

Sensorless problems

A vaccum agent knows all effects of its actions, but no sensors. The 8 possible states of vaccum world

1. 
2. 
3. 
4. 
5. 
6. 
7. 
8. 

- No sensor
- Initial state (1, 2, 3, 4, 5, 6, 7, 8)
- After action [RIGHT] the state (2, 4, 6, 8)
- After action [SUCK] the states [4, 8]
- After action [Left] the States [3, 7]
- After action suck the state (8)

∴ [Right, suck, left, suck] can be cam without any sensor

Partial knowledge of states & actions.

lead to contingency problem.

If uncertainty is caused by actions of another agent: Adversarial problem.

When states and actions are unknown: Exploration problem

Eg) According to Murphy's law:

suck can dirty a clean carpet

Percept = [L, Dirty] = {1, 3}

[Suck] = {5, 7}

- [RIGHT] = {6, 8}

[Suck] in {6} = {8} ⟹ success

But, [suck] in {8} ⟹ failure

Belief-State: no fixed action sequences guarantee solution.

Relax requirement:

[suck, RIGHT, if [R, dirty] then suck]

Contingency planning