UNIT-I

→ Introduction - The foundations of AI, History of AI
- What is AI? (003)
- AI characteristics
- foundation areas of AI (006-007)
- The History of AI (008)
- Applications of AI (009)
- Intelligent system or Diff types of AI (003-006)

→ Intelligent agents
- Agents and Environments (010)
- Good Behavior: Rationality (011)
- The Nature of Environment (012)
- The structure of Agents (014)
  1) Simple Reflex agents (014)
  2) Model Based Reflex Agents (015)
  3) Goal Based Agents. (016)
  4) Utility Based Agents. (017)

→ Solving Problems by Searching
- Problem solving agents (18)
- Production System (19)
- State Space Search (18)
- Control Strategies (20)
- characteristics of problems.
- Uninformed or Exhaustive serches:
  1) Breadth first Search (027)
  2) Depth first search (30)
  3) Depth first Iterative Deepning (32)
  4) Bidirectional Search (33)
  5) Uniform cost Search (29)
  6) Comparision of all searches (34)
  7) Analysis of Search Methods (34)
- Heuristic Search / Informed Search (035)
  1) Greedy Best first search (040)
  2) A* Search (042)

3) Branch and Bound search (29) (Uniform Cost search)
4) Hill Climbing (37)

Problems :
1. Water-Jug Problem (021)
2. The travelling Salesman Problem
3. Missionaries and cannibals problem (026)
4. 8 puzzle problem (025) (036)
5. M-C-W-G problem. (026)

→ Adversarial Search (048)
- Games (048)
- Optimal Decision in games : Minimax Algorith (049)
- Alpha-Beta Pruning (053)

→ Constraint Satisfaction Problem (057)
- Definition (057)
- Crypt Airthmatic puzzle (059)
- Map coloring (60)

DATE

What is Artificial Intelligence.

AI = Artificial + Intelligence
↓ ↓
non-natural ability to understand, think and learn.

→ "AI is the science and engineering of making intelligent machines especially intelligent computer program."
– John McCarthy.

or

→ AI as the study of making computers do things intelligently. Even now computers and systems are doing the work but that is all pre-programmed, the difference we want to make is – to enable them to do the work intelligently.

→ what do we mean by doing the work intelligently is:

Rational Thinking, Developing Perception, Taking an Actio

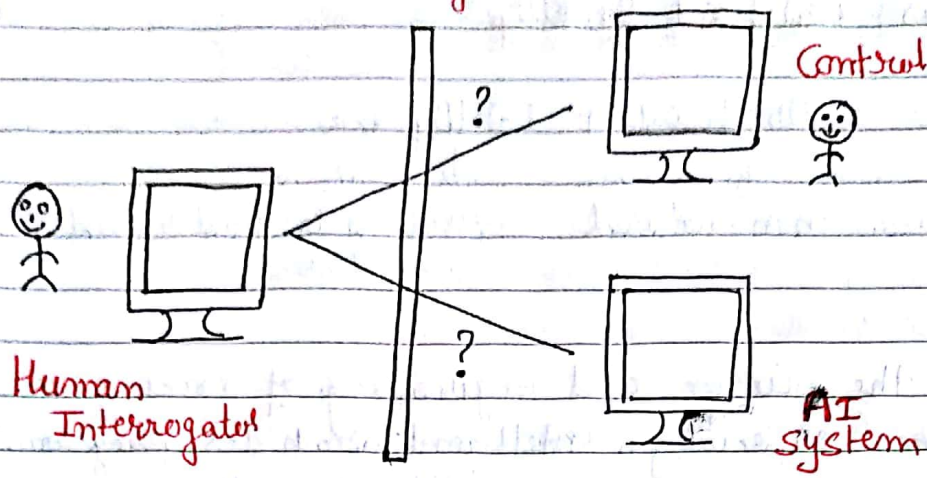
Categorization of Intelligent System

① Acting Humanly: The turing test approach (The Intimation Game)

→ The turing test, proposed by Alan Turing (1950), was designed to provide a satisfactory operational definition of intelligence.

→ A human interrogater was connected to two terminals and asked a series of questions.
One of the terminal had another human making the responses and other used a computer program to make the response.

Turing Test



→ If the human interrogater could not distinguish b/w human responder and the program response then the software was said to be intelligent.

The Computer would need to possess the following capabilities:

1) Natural language processing : to communicate successfully in english.

2) Knowledge Representation : to store what it knows or hears.

3) Automated Reasoning : to use the stored information to answer questions and to draw new conclusions.

4) Machine learning : to adapt to new circumstances and to detect and extrapolate patterns.

5) Computer Vision : to perceive objects

6) Robotics : to manipulate objects and move about.

→ The very first intelligent system ELIZA passed the turing test. Elize was a program that conversed with user in english.

(2) Thinking Humanly : The Cognitive Modelling Approach.

→ If we are going to say that a given program thinks like a human, we must have some way of determining how human think or actual working of human mind.

→ There are 3 ways to do this
1) through introspection – trying to catch our own thoughts as they go by.
2) through psychological experiments – observing a person in action.
3) through brain imaging – observing the brain in action.

→ Cognitive Science gives us how we think. It brings together computer models from AI and experimental techniq from psychology to construct precise and testable theories of the human mind.

→ Example: Conduct experiments with people, try to reverse engineer how we reason, learning, remembe & predict.

→ Neural Network is a computing model for processing information similar to brain.

(3) Thinking Rationally : The laws of thought approach.

→ Represent facts about the world via Logic.
→ Use logical inferance as a basis for reasoning about these facts.
→ Eg: John is a human and all human are mortal; therefore John is a mortal
There laws of thought were suppose to govern the operation of mind; their study initiated the field called logic.

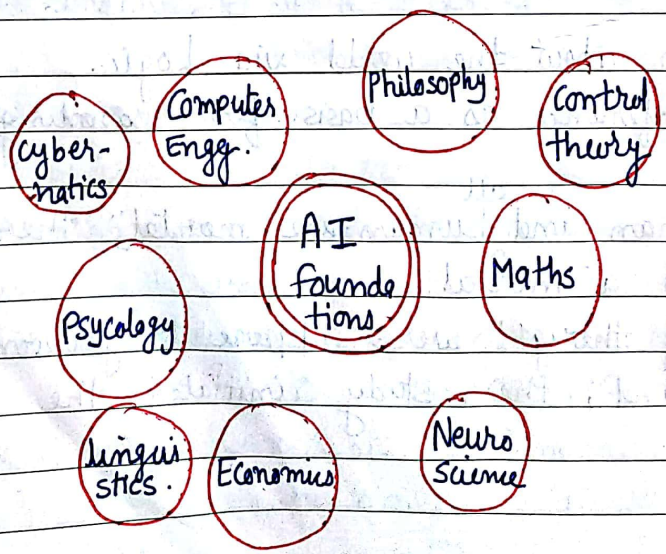
(4) Acting Rationally : The Rational Agent Approach.

→ An agent is just something that acts.
→ Rational behaviour we mean by doing the "Right thing" even if the method is illogical, the observed behaviour must be rational.

→ In the "laws of thought" approach to AI, the emphasis was on correct information inference.
→ Making correct inferences is something being a part of rational agent but it is not all about the rationality; in some situations, there is no provably correct thing to do, but something must still be done.

→ The rational agent approach has 2 advantages:
1) It is more general than 'laws of thought' approach because correct inference is just one of several possible mechanism to achieve rationality.
2) It is easy to adapt scientific development than are approaches based on human behavior or thought.

The Foundations of Artificial Intelligence



DATE

→ **Mathematics**: AI systems use formal logic methods and Boolean logic, analysis of limits to what can be computed, probability theory, uncertainty that forms the basis for most modern approaches to AI, fuzzy logic.

→ **NeuroScience**: This science of medicine helps in studying the functioning of brain. In early studies, injured and abnormal people were used to understand what parts of brain work. Now recent studies use accurate sensors to correlate brain activity to human thought.

→ **Control theory & Cybernatics**: How can artifacts operate under their own control?

Machines can modify their behaviour in response to the environment (sense / action loop)

→ **linguistics**: How does language relate to thought?

Speech demonstrate so much of brain intelligence. Modern linguistics and AI were born on the same time and grew up together, intersecting in a hybrid field c/c computational linguistics or Natural language processing.

→ **Psychology**: How do humans and animals think & act?

for this, we have cognitive psychology, which views brain as an information processing unit. We also have cognitive science to do this.

→ **Computer Engineering**: How can we build an efficient computer?

For AI to succeed, we need to things: intelligence and an artifact. The computer has been artifact of choice

→ **Philosophy**: Can formal rules be used to draw valid conclusion?
How does the mind arise from physical brain?
Where does knowledge come from?
How does knowledge lead to action?

→ **Economics**: How should we make decisions so as to maximize payoff?

How should we do this when others may not go along?
How should we do this when payoff may be far in the future?

The History of AI

Applications of AI

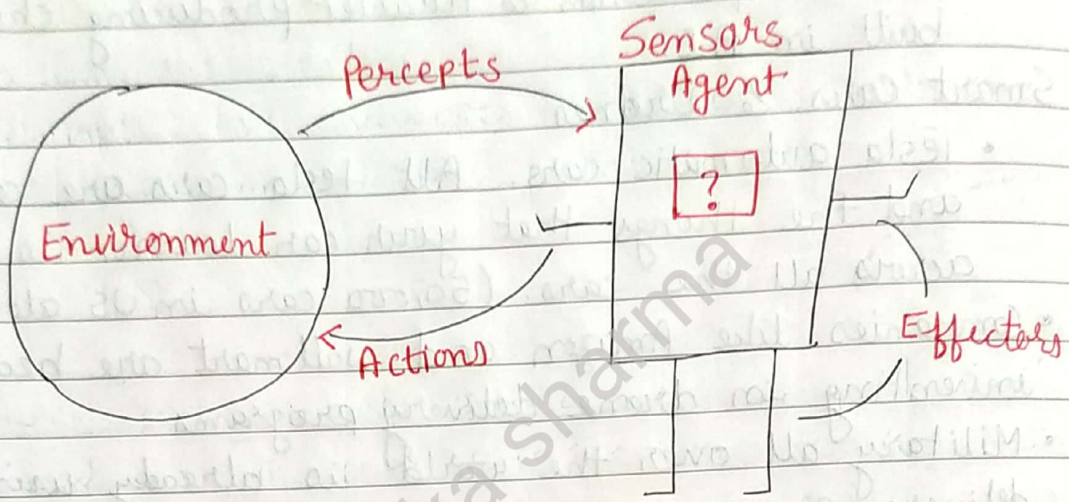
1. SmartPhones
   - Built-in Smart Assistants (Alexa, Siri, Google Assi.)
   - Portrait mode in the camera. (Google Pixel 2)
   - manufacturers are including AI in their smart phones – Qualcomm & Huawei producing chips with built in AI.
2. Smart Cars & Drones
   - Tesla automatic cars. All tesla cars are connected and the things that your car learns is shared across all the cars. (50,000 cars in US alone)
   - Companies like Amazon and wallmart are heavily investing in drone delivery programs
   - Military all over the world is already using drone delivery program.
3. Social Media feeds. (Twitter, facebook, insta)
   - from the feeds that you see in your timeline to the notifications that you receive from these Apps, everything is curated by AI.
   - The sole purpose of AI here is to make the apps so addictive that you come back to them again & again.
4. Music & Media Streaming Services.
   - whether you are using spotify, Netflix or youtube AI is making the decisions for you.
5. Video Games
   - earliest adopters of AI.
   - OpenAI5, Prolevel Dota 2

Intelligent Agents

DATE

What are agents & environment

An Agent is anything that can perceive its environment through sensors and acts upon that environment through actuators / effectors.



→ A human agent has sensory organs such as eyes, ears, nose, tongue, and skin parallel to the sensors and other organs such as hands, legs, mouth for effecters.

→ A robotic agent replaces camera and infrared range finders for the sensors and various motors and actuator for effectors.

→ A s/w agent receives key strokes, file contents and n/w packets as sensory I/p and acts on the enviornme by displaying on the screen.

Agent Terminology

→ **Percept**: It is agent's perceptual inputs at any given instant.

→ **Percept Sequence**: It is the complete history of every thing the agent has ever perceived.

→ **Agent function**: An agent's behaviour is described by agent function that maps any given percept sequence to an action.

DATE

→ **Agent Program:** Given an agent to experiment with, we can, construct table by trying out all possible percept sequence and recording which actions the agent does in response to it. This table is external characterization of the Agent. But Internally, it is implemented by Agent Program.

→ The agent function is an abstract mathematical description; The agent program is a concrete implementation, running within some physical system.

The Concept of Rationality

→ **Rationality** is nothing but status of being reasonable, sensible and having good sense of judgment.

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

1) The performance measure that defines the criteria of success.
2) The agent's prior knowledge of the environment.
3) The actions that the agent can perform.
4) The agent's percept sequence to date.

Definition of

→ **Rational Agent :** For each possible percept, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence & whatever the built-in knowledge the agent has.

or

An Ideal rational agent is the one, which is capable of doing expected actions to maximize its performance measure on the basis of :

- It's percept sequence
- It's built-in knowledge Base

The Nature of Environment

Task Environment: The problem the agent solves is characterized by Performance, Measure, Environment, Actuators and Sensors (PMEAS) or (PEAS)

This PEAS is known as task environment.

Example: PEAS description of the task environment for an automated taxi:

Agent Type	P.M	E	A	S
Taxi Drivers	safe, fast legal, comfortable trip, maximize profit	Roads, other traffic, pedestrians customers.	steering accelerator brake signal horn display	Camera Sonar speedometer GPS accelerometer engine sensors

Properties of Task Environment

1) **Fully Observable / Partial Observable / Unobservable**

- If an agent's sensors give it access to the complete state of the environment at each point in time, then fully observable
- An environment might be partially observable because of noisy and inaccurate sensors or because part of the states are simply missing from the sensor data.
- If the agent has no sensors at all then it is unobservable

2) **Sigle agent / Multiagent**

- an agent solving a crossword - single
- an agent playing a chess - two agent or multiagent

3) Competitive | Cooperative

Chess is a multiagent competitive environment.
(Maximize our performance by minimizing opponents performance)
In the taxi driving env: avoiding collisions maximizes the performance measure of all agents. So it is partially cooperative.

4) Deterministic | Stochastic. | Non-Deterministic

If the ~~current~~ next state of the environment is completely determined by the current state and the action executed by the agent, then env. is Deterministic otherwise it's stochastic.
Non-Deterministic env. is one in which actions are characterized by their possible outcomes, but no probabiliti are attached to them.

5) Episodic | Sequential

In an episodic task environment, the agent experience is divided into atomic episodes. In each episode agent performs a single action which does not depend on previous action or next action. (Eg: Spot defective parts on an assembly line)
In sequential env., the current decision could affect all future decision. (chess)

6) Static | Dynamic : If the env can change while an agent is deliberating, it is dynamic env otherwise static.
Taxi deriving: Dynamic, cross word - static.

7) Discrete | Continuous : Chess env has finite no of distinct states and discrete set of percept and action.
Taxi driving is a continuous state and cont. time problem.

The Structure of Agents

→ The job of AI is to design an agent program that implements the agent function. - the mapping from percepts to action.
This agent program will run on some sort of computing device with physical sensors and actuators c/d as architecture.

[agent = architecture + Program]

→ There are four basic kind of agent programs :
1. Simple Reflex Agent
2. Model Based Reflex Agent
3. Goal Based Agent
4. Utility Based Agent.

1. Simple Reflex Agent

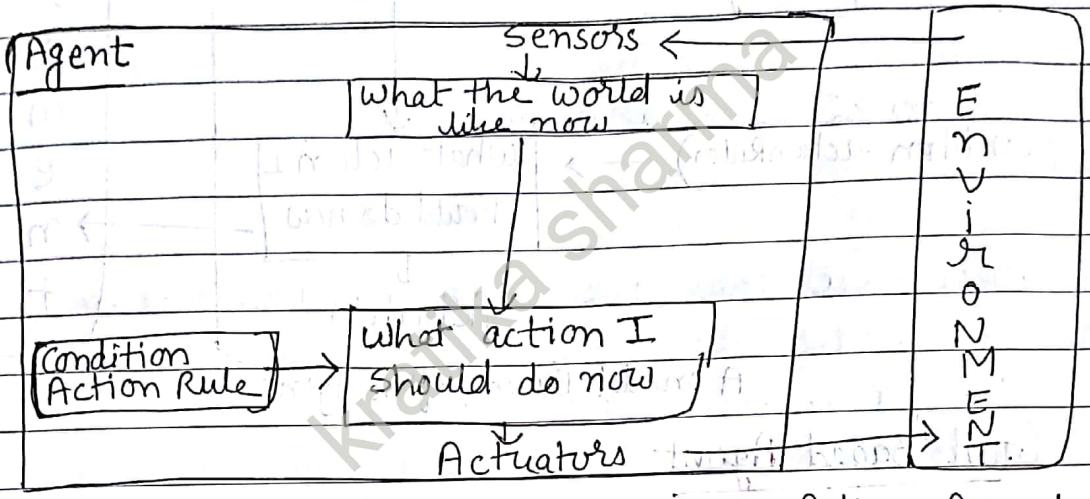
→ These agents select actions on the basis of the current percept, ignoring the rest of percept History.

→ The agent function is based on " Condition - Action Rule"
A condition action rule is a rule that maps a state i.e condition to an action.
If the condition is true, then the action is taken else not.

→ This agent function only succeeds when the environment is fully observable, if it is partially observable, infinite loops are often unavoidable.

DATE

→ Problem with simple Reflex Agents are
1) Very limited intelligence
2) No knowledge of non perceptual parts of state.
3) usually too big to generate and store.
4) If there occurs any change in the env, then the collection of rules need to be updated.

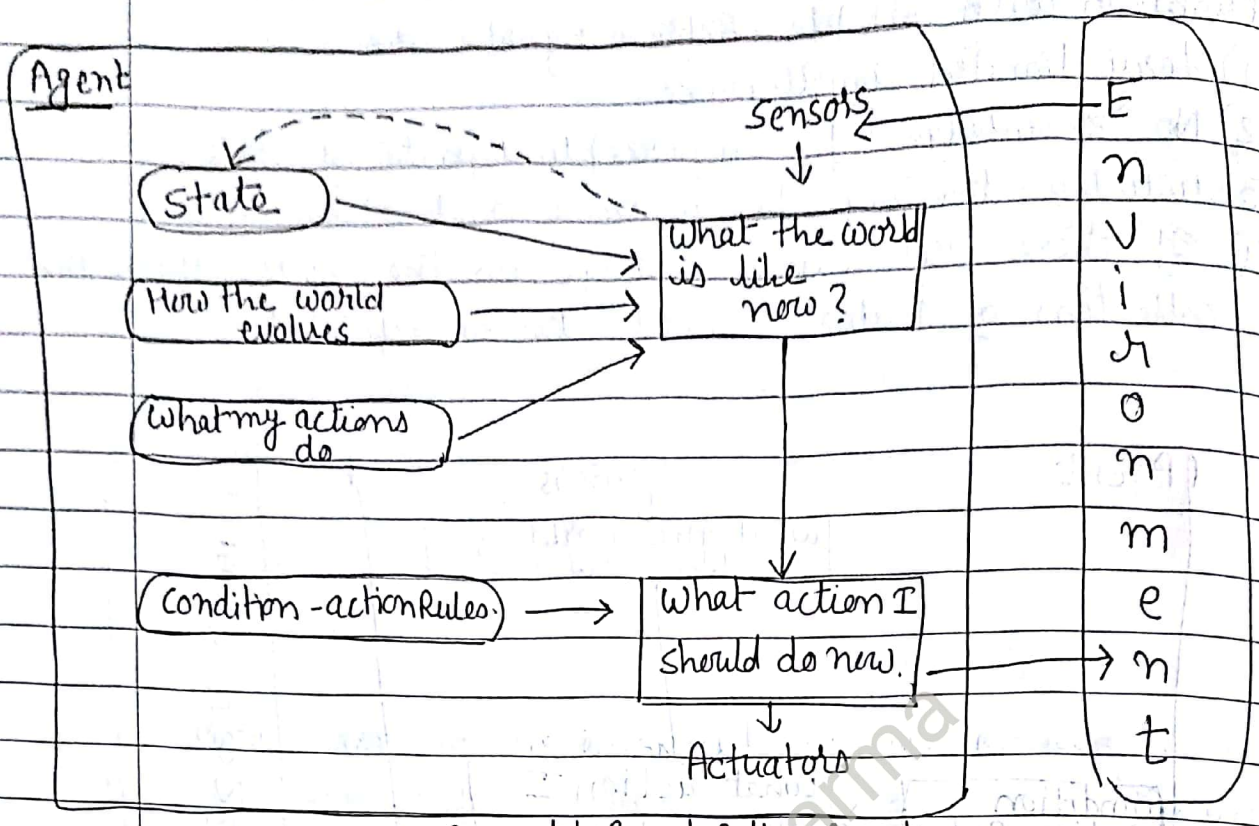


Schematic diagram of simple Reflex Agent

② Model Based Reflex Agent.

→ It works by finding a rule whose condition matches the current situation.
→ A model based agent can handle partially observable environments by use of model about the world.
→ The agent has to keep track of internal state which is adjusted by each percept and that depends on the percept history.
→ The current state is stored inside the agent which maintains some kind of structure describing the part of the world which can not be seen.

→ Updating the state require the info. about:
- How the world evolves in-dependently from the agent &
- How the agent action affects the world.



A model Based Reflex Agent.

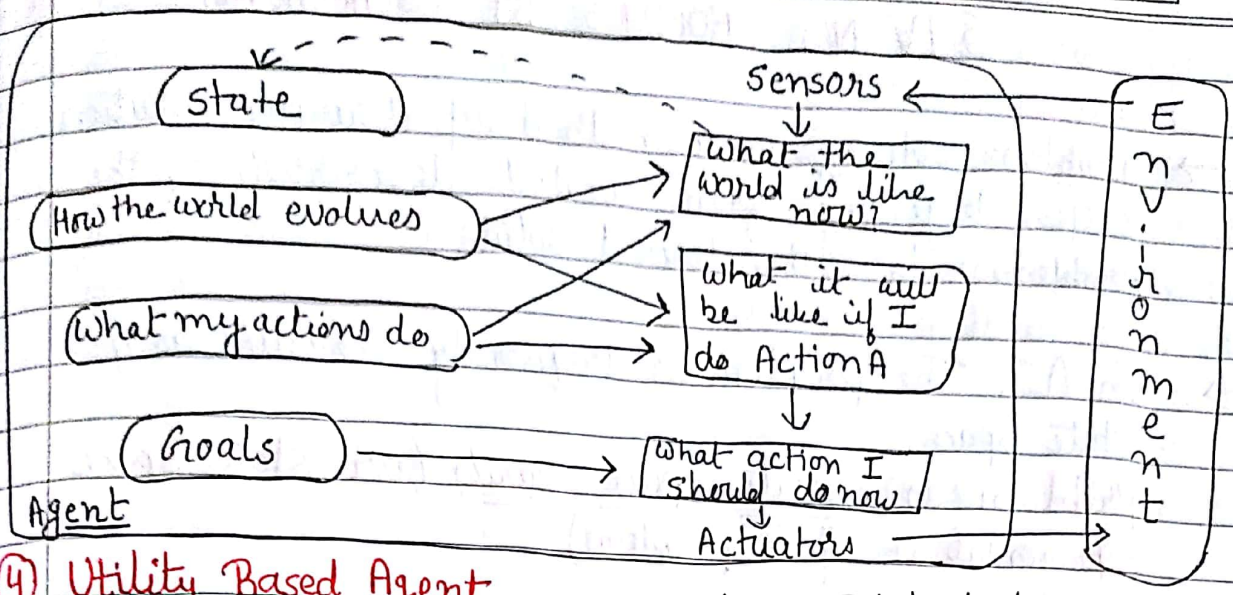
③ Goal Based Agent

→ This kind of agent take decision based on how far they are currently from their goal.

→ There Their every action is intended to reduce its distance from goal. This allows the agent a way to choose among multiple possibilities, selecting the one which reaches a goal state.

→ The knowledge that supports its decision is represented explicitly and can be modified, which makes these agents more flexible.

→ They usually require search & planning. The goal based agent's behavior can easily be changed.



(4) Utility Based Agent

in above diagram instead of goals draw

Utility → How happy I will be in such a state

→ The agents which are developed having their enel ures as building blocks are cld utility based agents. (action ?)

→ When there are multiple possible alternaties, then to decide which one is best, utility based agents are used.

→ They choose actions based on a preference (utility) for each state.

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

→ Agent happiness should be taken into consideration.

→ Utility describes how "happy" the agent is. Because of the uncertainty in the world, a utility agent chooses the action that maximizes the expected utility.

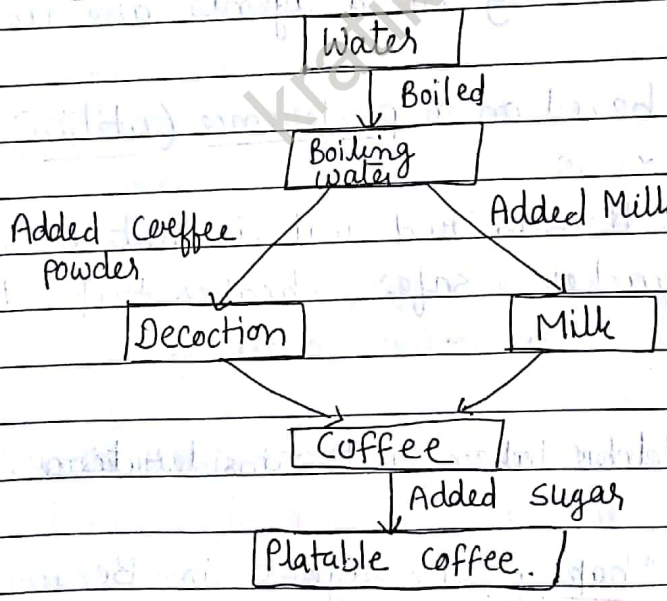
→ A utility function maps a state onto a real number which describes the associated degree of happiness.

SOLVING PROBLEMS BY SEARCHING.

→ Problem solving is a method of deriving solution steps begining from initial description of the problem to the desired solution.

→ In AI, The problem is frequently modelled as a state space.
Problem as a state space search (or state space representation of a problem)

→ A set of all possible states for a given problem is known as the state space of the problem.
Or
→ State space is a set of all possible states from start (initial) to goal state.



→ Basically to do a search process, the following are needed:
① The final state description of the problem.
② A set of legal operators that changes the state.
③ The final or the goal state.

→ Searching can be defined, as a sequence of steps that transforms the initial state to the goal state.

DATE

→ There is a difference b/w the state space search used in AI and the conventional C.S. search method. In the case of the former, the states of the graph (search space) are generated and as they are explored for the solution path and discarded thereafter, whereas in the traditional search method, seaches are applied to the existing graphs.

→ The typical state space graph for solving problem is too large to be generated and stored in memory.

→ The two types of problem solving methods that are generally followed include general purpose and special purpose methods.

General Problem Solving
- Production System
- state space search
- control strategy.

PRODUCTION SYSTEM

"Production Systems provide structure which describe and perform search process."

A production system Consists of :

① A Set of Rules (Productions or production Rule)

$C_i \rightarrow a_i$

Condition part action part

② One or more knowledge Base :
Consisting of suitable and necessary information for particular task.

③ Control strategy
It specify the order in which the rules will be Compared to the database and a way solving the Conflicts

classmate

that arise when several rules match at once.

④ A rule applies
Ex: Expert System shell, general problem solving architecture.

Control Strategies

→ It is a strategy by which we come to know which rule is to be applied next during the process of reaching for a solution to a problem.

→ Control strategy help us to overcome the abnormal situations, where there are more that one rule or fewer than one rule will have left sides (condition) match the current state.

→ A good control strategy must have the following characteristics (requirment):

① It should cause Motion : If there is no motion that means there is no chang in state and if state is not changed, then we will never proceed from initial to goal state.

② It should be systematic : A control strategy which cause motion but not systematic, One can reach to a solution but it may pass from one state many times or may use many more steps than are necessary.

→ Some Widely Used Control strategies are:
① Uninformed or Exhaustive or Systematic uninformed exhaustive Searches (DFS, BFS, DFID, Bidirectional)
② Informed Or Heuristic Searches
(Greed Best first search, A*, Branch & Bound Hill climbing)

DATE

Water Jug Problem

Problem Statement : We have two jugs, a 5 gallon(5-g) and 3 gallon (3g) with no measuring marker on them. There is endless water supply through tap. Our task is to get 4 gallon of water in 5-g jug

Solution : The state for this problem can be described as the set of ordered pairs of integers (X, Y) such that X = 0, 1, 2, 3, 4, 5 no. of gallons of wates in 5g jug

Y = 0, 1, 2, 3 no. of gallons of wates in 3g jug.

Here start with (0, 0) both jugs are empty
Goal state (4, N), N <= 3, that is first jug must have 4g and second jug can have any value.

Production Rules for W-J Problem

Rule No.	left of Rule (condition)	Right of Rule (Transition)	Description.
1.	(X,Y | x<5)	(5, Y)	Fill 5 g jug
2.	(X,Y | x>0)	(0, Y)	Empty 5 g jug
3.	(X,Y | y<3)	(X, 3)	Fill 3 g jug
4.	(X,Y | y>0)	(X, 0)	Empty 3 g jug
5.	(X,Y) | x+y ≤ 5 and y>0)	(x+y, 0)	Empty 3g into 5-g
6.	(X,Y | x>0 and x+y ≤ 3)	(0, x+y)	Empty 5g into 3g
7.	(x,y | x+y ≥ 5 and y>0)	(5, y-(5-x))	pour water from 3g into 5g.
8.	(X,Y | x>0 and x+y ≥ 3)	(x-(3-y), 3)	pour water from 5g until 3g is full

classmate

PAGE 021

Solution Path 1 for W-J Problem

Rule applied	5 g jug	3 g jug	step no.
Start state	0	0	1
1.	5	0	2
8.	2	3	3
4	2	0	4
6	0	2	5
1	5	2	6
8	4	3	
Goal state	4	-	

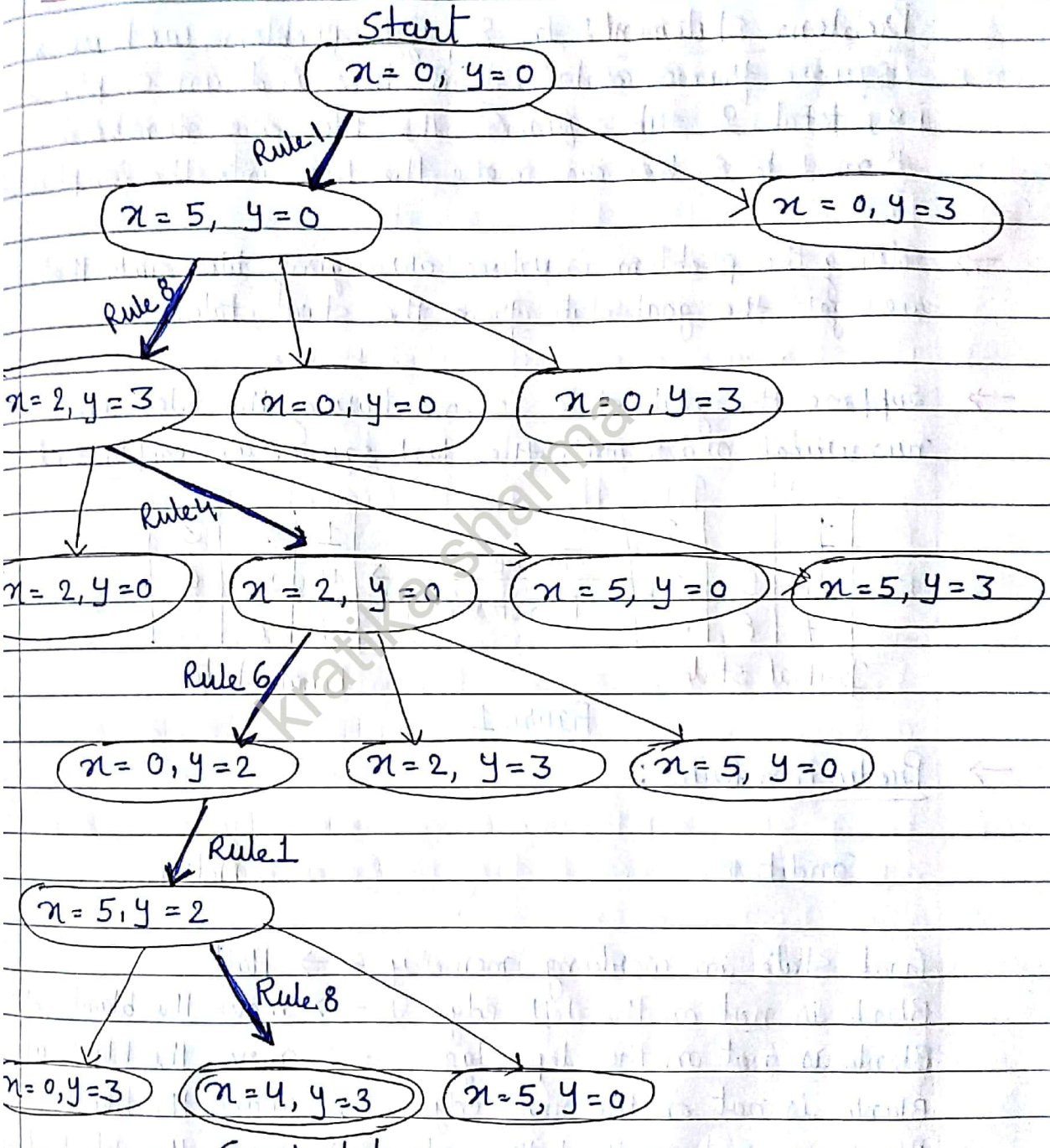
→ It should be noted that there may be more than one solution for a given problem.

Solution Path 2 for W-J Problem.

Rule Applied	5 g jug	3 g jug	step no.
Start state	0	0	
3	0	3	1.
5	3	0	2
3	3	3	3
7	5	1	4
2	0	1	5
5	1	0	6
3	1	3	7.
5	4	0	8.
Goal state.	4	-	

DATE

State Space Representation of Water-jug problem.



Each state in state space representation is denoted by (x, y).
Here x is a 5-g. jug while y is 3 g jug.
Each state in a circle shows a resultant state after application of appropriate rule on that state.
Initial state is represented as start while final state is enclosed with in double circle.

DATE

8 Puzzle Problem

Problem Statement: In 8 puzzle problem, we have a square frame containing 8 tiles and an empty slot i.e, total 9 sub squares. The tiles are numbered from 1 to 8. We can move the tiles into the empty slot

→ Solving the problem involves arranging tiles such that we get the goal state from the start state.

→ Suppose the goal state is one having the tiles in numerical order with the last square an empty slot.

1		2
4	5	3
7	8	6

Initial State

To be Transformed →

1	2	3
4	5	6
7	8	

Final State

Figure 1.

→ **Production Rules:**

Condition	Action
Goal state in working memory	→ Halt
Blank is not on the left edge	→ move the blank left
Blank is not on the top edge	→ move the blank up
Blank is not on the right edge	→ move the blank right
Blank is not on the bottom edge	→ move the blank down

Control Regime:

1. Try each Production in Rule
2. Do not allow loop.
3. Stop when goal is found.

→ State Space representation

Initial & goal states are shown in figure 1.

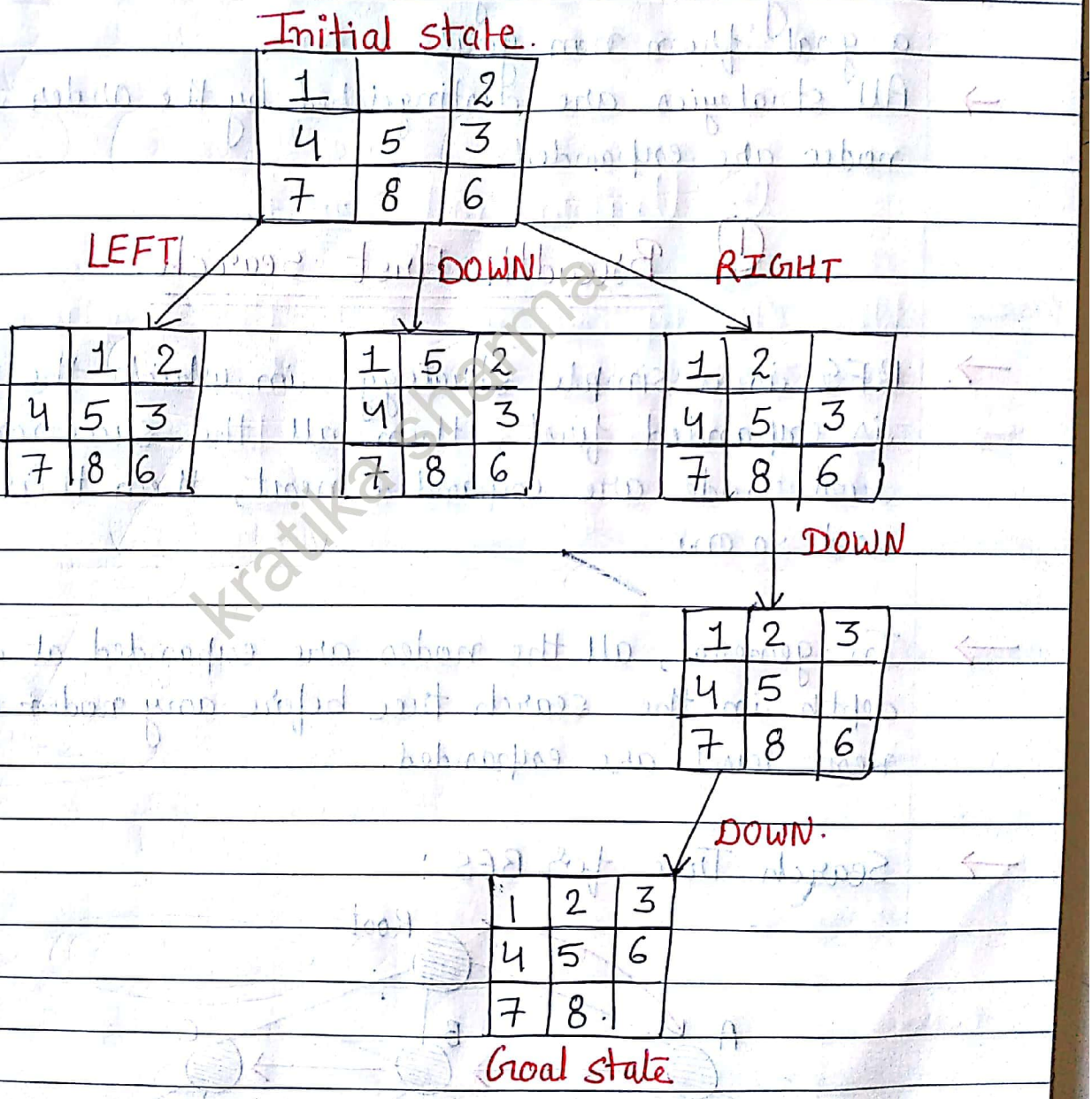
operators for this problem are:

UP - If the hole (Blank) is not touching the top frame, move it up.

DOWN - If the Blank is not touching the bottom, move it down

LEFT - If the Blank is not touching the left frame, move it left

RIGHT - If the blank is not touching right frame, move it right



Note: Missionaries and cannibals problem
Man, Cabbage, Wolf and goat problem
Write down Production Rules and draw state space representation of these two problems.

UNINFORMED OR EXHAUSTIVE SEARCHES OR BLIND SEARCH.

→ The term uninformed or blind or exhaustive means that the strategies have no additional information about states beyond that provided in the problem definition.

→ All they can do is generate successors and distinguish a goal from non goal.

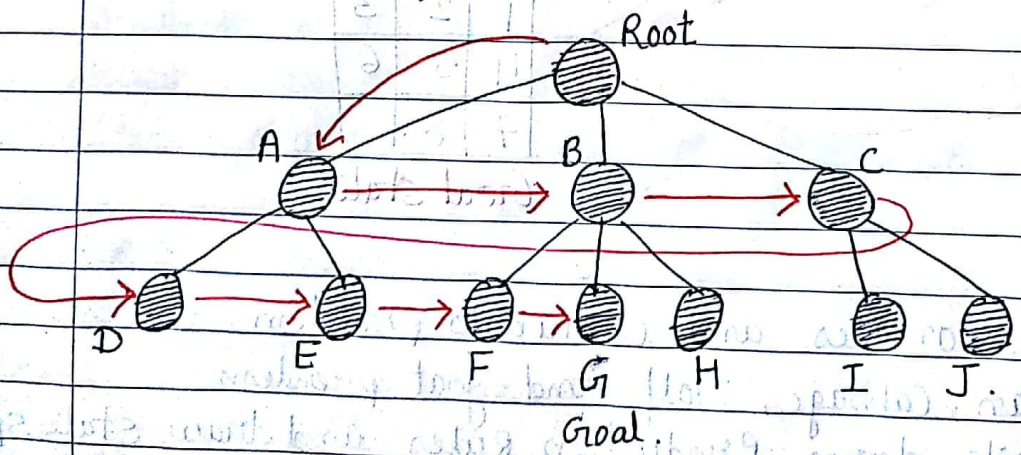
→ All strategies are distinguished by the order in which nodes are expanded.

① Breadth First Search.

→ BFS 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.

→ In general, all the nodes are expanded at a given depth in the search tree before any nodes at the next level are expanded.

→ Search Tree for BFS :



→ This search is implemented using 2 list c/d OPEN & CLOSED. The OPEN list contains those states that are to be expanded & CLOSED list keeps track of states already expanded.

OPEN - as queue (FIFO)
CLOSED - as stack (LIFO)

→ **Algorithm :**

Input : start & goal states.
Local Variables : OPEN, CLOSED, STATE-X, SUCCs, FOUND.
Output : Yes or No
Method :
→ initialize OPEN list with START & CLOSED = ϕ;
→ FOUND = false;
→ while (OPEN $\neq \phi$ and FOUND = FALSE) do
{
- remove the first state from OPEN and call it STATE-X;
- put STATE-X in the front of CLOSED list (maintained stack)
- If state-X = GOAL then FOUND else
  - perform EXPAND operation on STATE-X producing a list of SUCCs;
  - remove from successors those states, if any, that are in the closed list.
  - append SUCCs at the end of the OPEN.

}
→ if FOUND = true then return Yes else return No
→ stop.

→ **Time Complexity :** Imagine searching a uniform tree where every state has b successors.
→ The root of the tree generates b nodes at the first level, each of which generates b more nodes, for a total b^2 at the second level. Each of these generates b more nodes, getting b^3 nodes at third level and so on.
→ Now suppose that the solution is at depth d.
Total no. of nodes generated $b + b^2 + b^3 + \dots + b^d = O(b^d)$

DATE

→ **Space Complexity** for any kind of graph search, which stores every expanded node in the explored set, the space complexity is always within a factor of b of the time complexity.
So the S.C is $O(b^d)$ total no. of expanded node in memory

→ **Problems with BFS (limitations)**
Memory requirements are a bigger problem for bFs than is the execution time.

(2) Uniform Cost Search.

→ When all step costs are equal, BFS is optimal because it always expands the shallowest unexpanded node.
→ By a simple extension, we can find an algorithm that is optimal with any step cost function.

DATE

Depth First Search.

DATE

Depth First Iterative Deepning

DATE

BIDIRECTIONAL SEARCH

INFORMED (HEURISTIC) Search Strategies.

→ Informed search is one that uses problem specific knowledge beyond the definition of problem itself.

→ It can find a solutions more efficiently than can an uninformed strategy.

→ Heuristic technique is a criteria for determining which among several alternatives will be the most effective to achieve the same goal.

→ This technique improves efficiency of a search process possibly by sacrificing claims of systematic & Completness.

→ It no longer guarentees to find the best solution but almost always finds a very good solution.

→ There are 2 types of heiristics :

① General purpose heuristic : It is useful in a wide variety of problem domain.

② Special purpose heuristic : It exploit domain specific knowledge to solve particular problem (informed search.)

Heuristic Function.

→ It is a function that maps from problem state description to measure of desirability, usually represented as number.

→ Which aspects of problem state are considered, how these states are evaluated, and weight given to the individual aspects are chosen in such a way that the value of the heuristics function at a given node in the search give as good as estimate as possible of whether that node is on the desired path to solution.

DATE

→ The purpose of heuristic function is to guide the search process in the most profitable direction by suggesting which path to follow first when more than one is available.

→ Heuristic function is denoted by $h(n)$ for node n.

evaluation function (which node to select next during search process) $f(n) = g(n) + h(n)$

Heuristic function for 8 puzzle problem

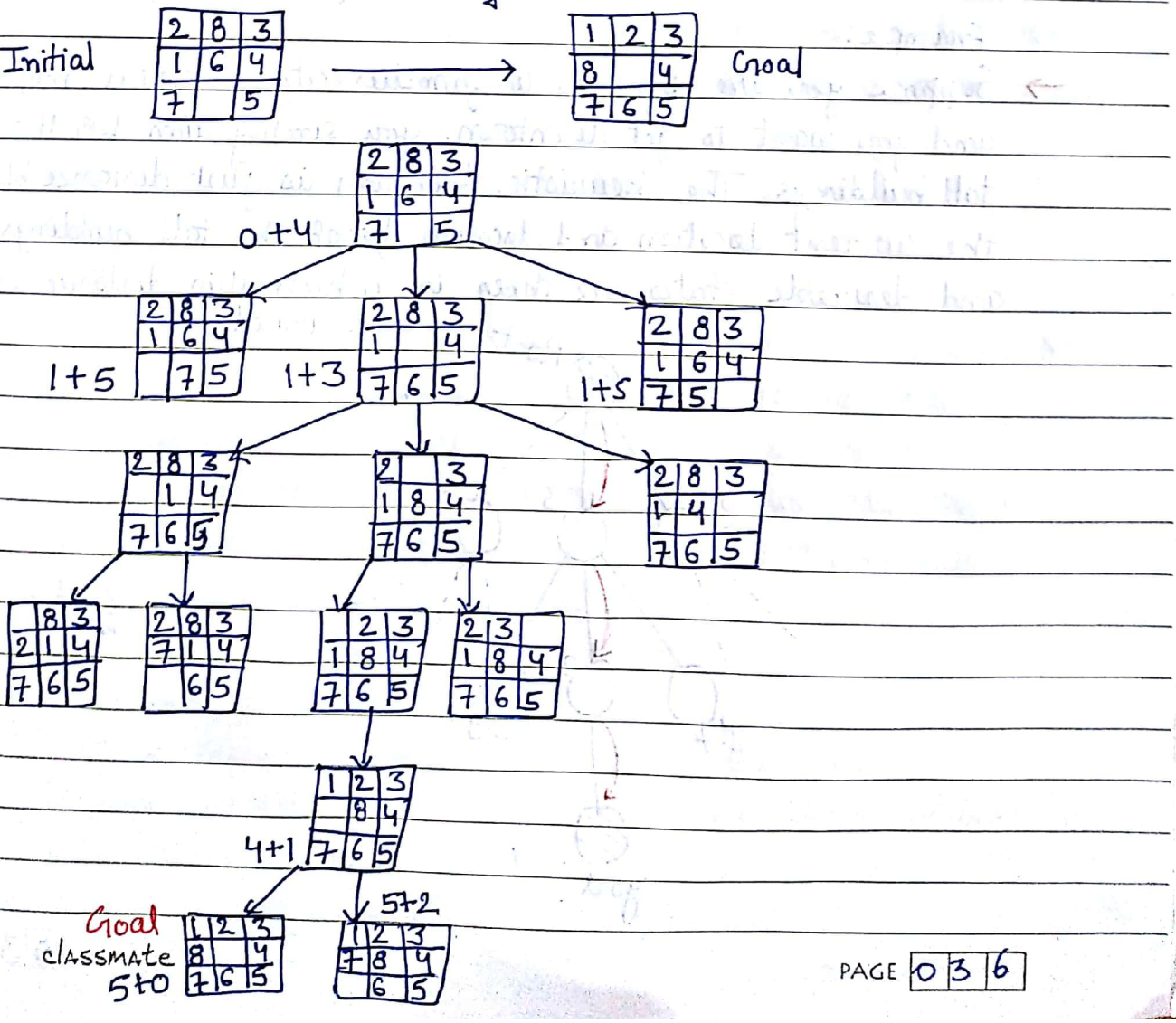
→ for 8 puzzle problem, we might try using the number of tiles out of place as a measure of goodness of a state description.

$\hat{f}(n)$ = no. of tiles out of place (compared to goal)

→ This example shows that, we need to bias the search in favour of going back to explore early path so we add 'depth factor' to

$\hat{f}(n) = \hat{g}(n) + \hat{h}(n)$

where $\hat{g}(n)$ is the estimate of "depth" of n in the graph, and $\hat{h}(n)$ is an heuristic evaluation of node n.



Hill Climbing procedure (Discrete optimization problem)

→ It is simply a loop that continually moves in the direction of increasing value - that is, uphill.

→ It terminates when it reaches a "peak" where no neighbor has a higher value.

→ The algorithm does not maintain a search tree, so the data structure for the current node need only record the state and the value of objective function.

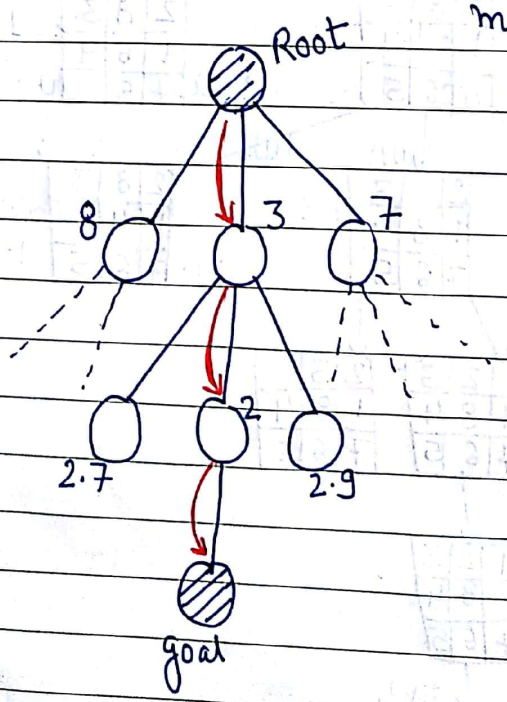
→ Hill climbing does not look ahead beyond the immediate neighbors of the current state.

→ It uses a simple heuristic function : the amount of distance the node is from the goal.

→ Hill climbing is often used when a good heuristic function is available but when no other useful knowledge is available.

Example:

→ Suppose you are in an unfamiliar city without a map and you want to get downtown. You simply aim for the tall buildings. The heuristic function is just distance b/w the current location and location of the tall buildings. and desirable states are those in which this distance is minimized.



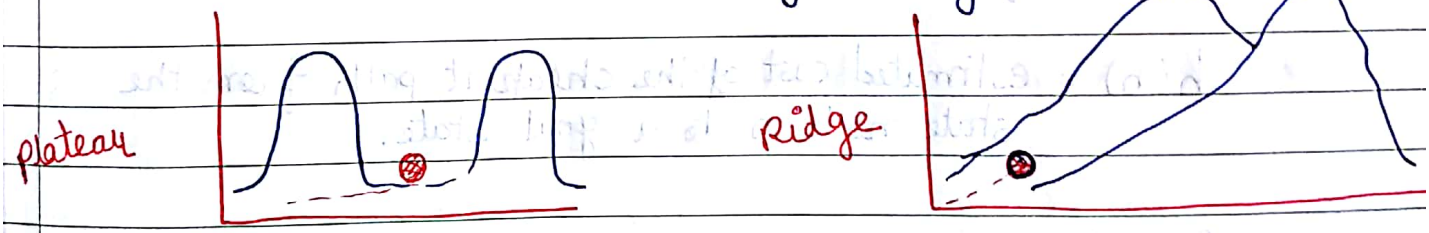
Problems With Hill climbing.

① **local Maxima** A local maximum is a peak that is higher than each of its neighboring states but lower than the global maximum.

At local maxima, all moves appear to take things worse. Local maxima are frustrating because they often occur almost with in sight of a solution c/d "foothills"

② **plateau** A plateau is a flat area of the search space in which a whole set of neighboring states have the same value.

On a plateau, it is not possible to determine the best direction in which to move by making local comparisons.



③ **Ridge** It is a special kind of local maxima. It is an area of the search space that is higher than surrounding areas and that itself has a slope. But the orientation of the high region, compared to the set of avaible moves and the direction in which they move, makes it impossible to traverse a ridge by single move.

Remedies to these problems:

① Backtracking for local Maxima.
② A Big jump is the solution to escape from plateau.
③ Trying different paths at the same time is the solution for circumventing ridge.

Best First Search

→ DFS is good because it allows a solution to be found without all competing branches having to be expanded. BFS is good because it does not get trapped on dead end paths.

→ One way of combining these two is to follow a single path at a time, but switch paths whenever some competing paths looks more promising than the current one does. This search is c/d Best First Search.

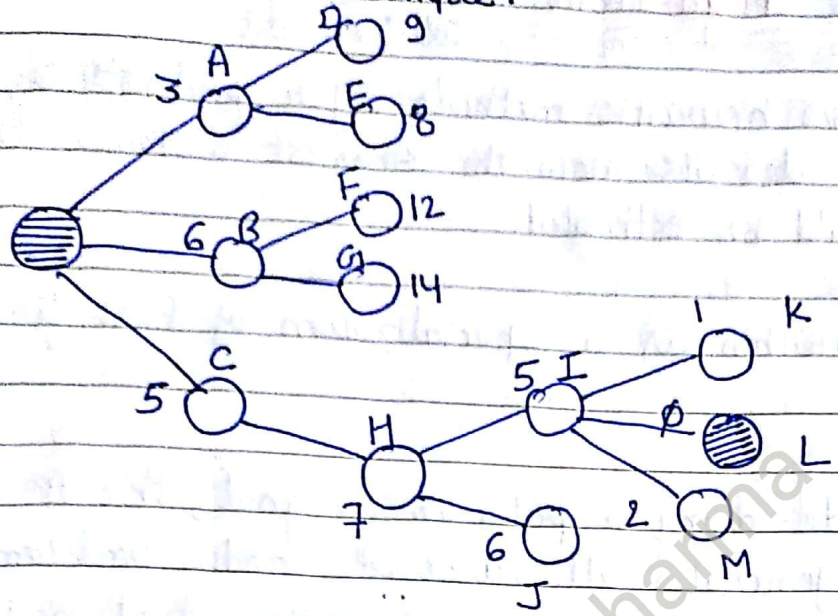
→ This search procedure is an evaluation function varient of Breadth first search.
The evaluation function f(n) is constructed as a cost estimate, so the node with lowest evaluation function is expended first.

→ Most best first search include as a component of f a heuristic function h(n)

h(n) = estimated cost of the cheapest path from the state node n to a goal state.

→ Greedy Best First Search tries to expend the node that is closest to the goal, on the grounds that is likely to lead a solution quickly.
Thus it evaluates nodes by using just the heuristic function; f(n) = h(n)

Just consider an example:



(A:3)
↳ evaluation function of node A.

Step	Node being Expanded	Children	Available Nodes	Node chosen.
1.	S	(A:3)(B:6)(C:5)	(A:3)(B:6)(C:5)	(A:3)
2	A	(D:9)(E:8)	(B:6)(C:5)(D:9)(E:8)	(C:5)
3	C	(H:7)	(B:6)(D:9)(E:8)(H:7)	(B:6)
4	B	(F:12)(G:14)	(D:9)(E:8)(H:7)(F:12)(G:14)	(H:7)
5	H	(I:5)(J:6)	(D:9)(E:8)(F:12)(G:14)(I:5)(J:6)	(I:5)
6	I	(K:1)(L:ø)(M:2)	(D:9)(E:8)(F:12)(G:14)(J:6)(K:1)(L:ø)(M:2)	search stops as goal is reached.

DATE

A* Algorithm: Minimizing the total estimated solution cost.

→ The previous heuristic methods offer good strategies but fail to describe how the shortest distance to a goal should be estimated.

→ The A* algorithm is a specialization of Best first Search.

→ At each node along a path to a goal, the A* algorithm generates all successor nodes and computes an estimate of the distance from the start node to a goal through each of the successors.

→ It then chooses the successor with the shortest estimated distance for expansion. The successors for this node are generated, their distances estimated and the process continues until a goal is found or search ends in failure.

→ A* can be used whether we are interested in finding a minimal cost overall path or simply any path as quickly as possible.

For this type of problem we need to maintain 2 lists of node types:

(i) OPEN - nodes that have been generated and had the heuristic function applied to them but which have not been expanded.

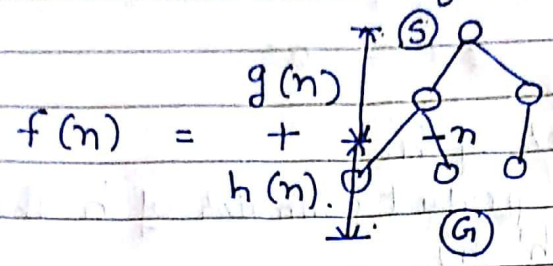
(ii) CLOSED - nodes that have been expanded and whose children are available.

It also needs 3 functions : $f(n) = g(n) + h(n)$

(i) $f(n)$ = It estimates the merits of each node we generate.

(ii) $g(n)$ = from start node to node n

(iii) $h(n)$ = Additional cost measure from node n to goal.



s = start

n = node.

G = Goal.

- Instead of calling F we will call it as F' to indicate that it is an approximation to a function F that gives the true evaluation of the node.
  Same we will call h' as an approximation of h.

How to decide $g(n)$?

$g(n)$ = how good node looks itself + how good the path to the node was.

★ with g, we will not always choose our next node to expand that appear to close to goal.
we choose $g(n)$ in such a way that we should get less cost.

Role of g

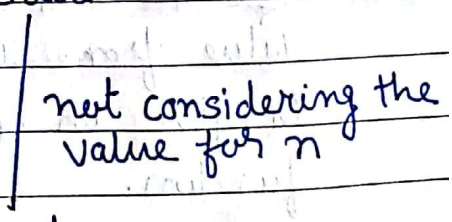
Two Imp features of $g(n)$

① If $g = 0$

$f(n) = 0 + h(n)$

$= h(n)$

that means we are not deciding n value, and directly traversing to the goal state.



Here in this case, we want to reach the goal state somehow. without bothering the path we have choosen and no. of steps it is taking.

When $g = 0$, $f(n) = h(n)$ we choose $h(n)$, that seems close to the goal state.

(2) $g = 1$ (constant)
in this case we want path involving fewest no. of steps.

Role of h

→ h' = estimator of h.
If h' is the perfect estimator of h then our goal ($h' = h$) will be covered without anymore searches.

better $h' = h$ (fastly we can reach towards goal)

→ If $h' = 0$, then search will be controlled by $g(n)$
also $g = 0$, then search will be random.
$g = 1$, then search will become Breadth first Search

→ If h' neither perfect nor zero ↓
Optimal solution by A* algorithm

Admissible Heuristic Or Admissibility of A*

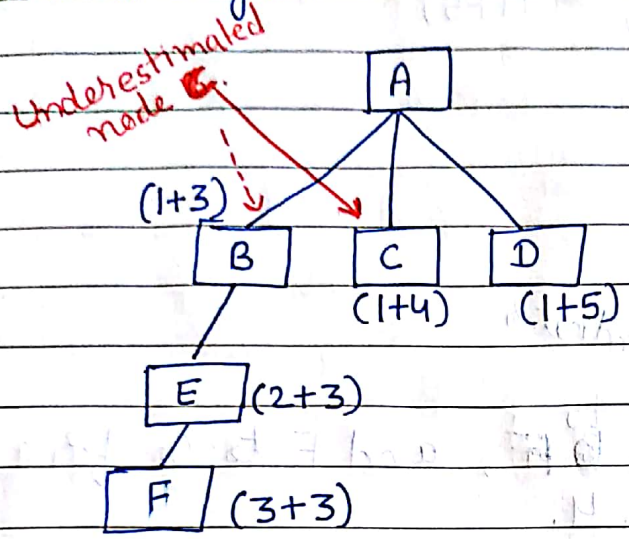
→ A search algorithm is admissible, if for any graph it always terminates in an optimal path from start state to goal state, if path exists.

If heuristic function h underestimates the actual value from current state to goal state, then it bounds to give an optimal solution and hence is called admissible function.

So we can say that A* always terminates with the optimal path In case h is an admissible heuristic function.

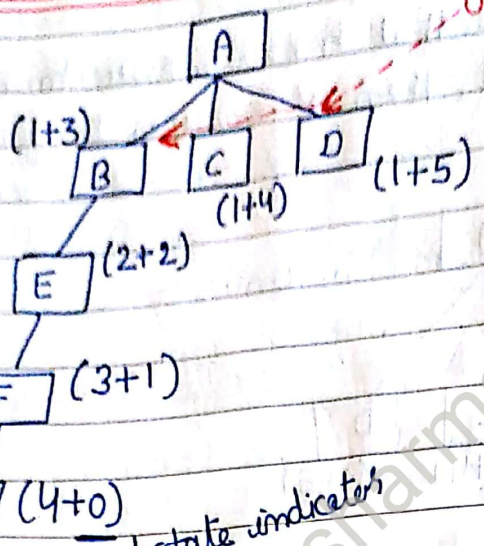
h' underestimates h (Underestimation)

If we can gurentee that h never overestimates actual value from current to goal, then A* ensures to find an optimal path to a goal.



Start node A is expanded to B, C and D with f values as 4, 5 & 6.
Here we are assuming that the cost of all arc is 1.
Note that node B has minimum f value, so expand this node to E which has f value as 5.
Since f value of C is also 5, we resolve in favor of E, the path currently we are expanding.
Now node E is expanded to node F with f value as 5
Clearly, expansion of node F is stopped as f value of C is the smallest.
We can see that by underestimating heuristic value, we have wasted some effort but eventually discovered that B was farther away than we thought.
Now we go back and try another path and will find the optimal path.

h' Overestimated h
(Overestimation)



we expand B to E, E to F, and F to G for a solution path of length 4.
But assume that there is a direct path from D to a solution giving a path length 2 as h value of D is also overestimated.
We will never find optimal solution by overestimating h, we may find some other worse solution without ever expanding D.

→ Identify the condition at which the A* will become -
(1) Breadth First Search (2) Best First Search
(3) No search is required.
According to A*

→ The heuristic function that estimates the merits of node we generate. This will enable the algorithm to search more promising path first.

→ Call this function f' (to indicate that it is an approximation to a function f that gives the true evaluation of the node).
and we know

$$f' = g + h'$$

→ The function g is a measure of the cost of getting from the initial state to the current node.

→ Note that g is not an estimate of anything it is known to be the exact sum of the cost of applying each of the rules that were applied along the best path to the node.

→ h' is an estimate of the additional cost of getting from the current node to a goal state.

→ The Combined function f', then represents an estimate of the cost of getting from the initial state to a goal state along the path that generated the current node.

→ There are conditions through which A* will become:

(1) Breadth First Search If the value of g is always 1.

(2) Best First Search If the value of h' is 0, then the search will be controlled by g

(3) No search is required If h' is a perfect estimater of h then A* will converge immedeatly to the goal without searching.

ADVERSARIAL SEARCH

Game playing

→ Competitive environment (chess, tic-tac-toe), in which the agent's goals are in conflict, giving rise to adversarial search problems – known as game

→ There were two reasons that games appeared to be a good domain in which to explore machine intelligence

1) They provide a structured task in which it is easy to measure success or failure
2) They did not obviously require large amount of knowledge. (seach from start to winning position)

→ What are the game playing strategies

→ Games can be classified as either a single person playing or multiperson playing.

→ For example, Rubik's cube and 8 tile puzzle are single persons game for solving such problems strategies like Best first or A* can be used.

→ While for solving problem, where 2 person play a game like chess or checkers can not be solved by above algorithms. As here each player tries to outsmart the opponent. Each has their own way to evaluating a situation.

→ The basic characteristic of the strategy must be look ahead in nature i.e. explore the tree for two or more levels downwards and choose the optimal one.

The basic methods available for game playing:

1. Miniman Strategy
2. Miniman strategy with α-β cut-offs.

DATE

Minimax Strategy

→ The minimax search procedure is a depth-first, depth limited search procedure.

→ The idea is to start at the current position and generate all possible successor positions.

→ Now by applying the static evaluation function (The SEF gives a snapshot of a particular move. More the SEF value, more is the probability for a victory) to those positions we can choose the best one.

→ After doing so we can back that value up to the starting position to represent our evaluation for it.

→ Here, one player is called maximizer and opponent is called minimizer.
Maximizer has to maximize its benefits and minimizer has to minimize its opponents benefit.
Also, we assume that first chance is of maximizer then of minimizer and so on.

→ Example:

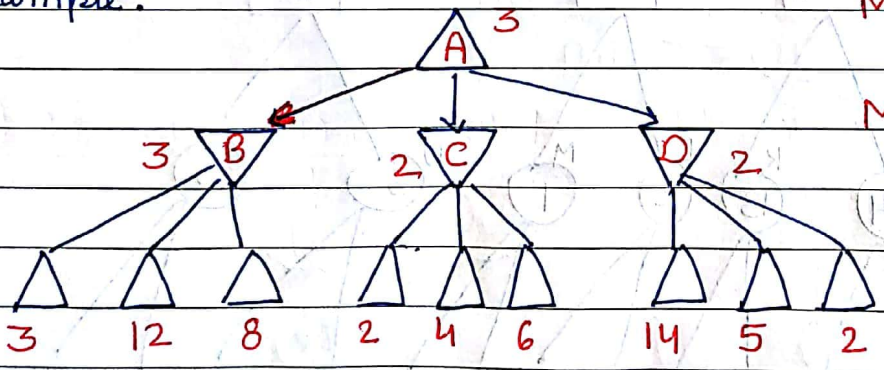


Figure (a)

→ A 2 ply game (ply - depth) tree. The △ nodes (or you can take □) are "MAX" nodes in which it is MAX's turn to move and ▽ (or ○) are "MIN" node. The terminal node's value shows the utility or static evaluation function value for MAX.

DATE

Figure (a)

→ The algorithm first recurses down to the 3 bottom left nodes and uses Utility function on them to discover that their values are 3, 12 & 8.

→ Then it takes the minimum of these values, 3, & returns it backed up value of node B.

→ A similar process gives 2 for C, 2 for D.

→ finally we take the maximum of 3, 2 & 2 to get value 3 for the root node.

Worst Case max value = $-\infty$

Worst case min value = ∞

△ □ - max

▽ ○ - min

Apply Mini Max:

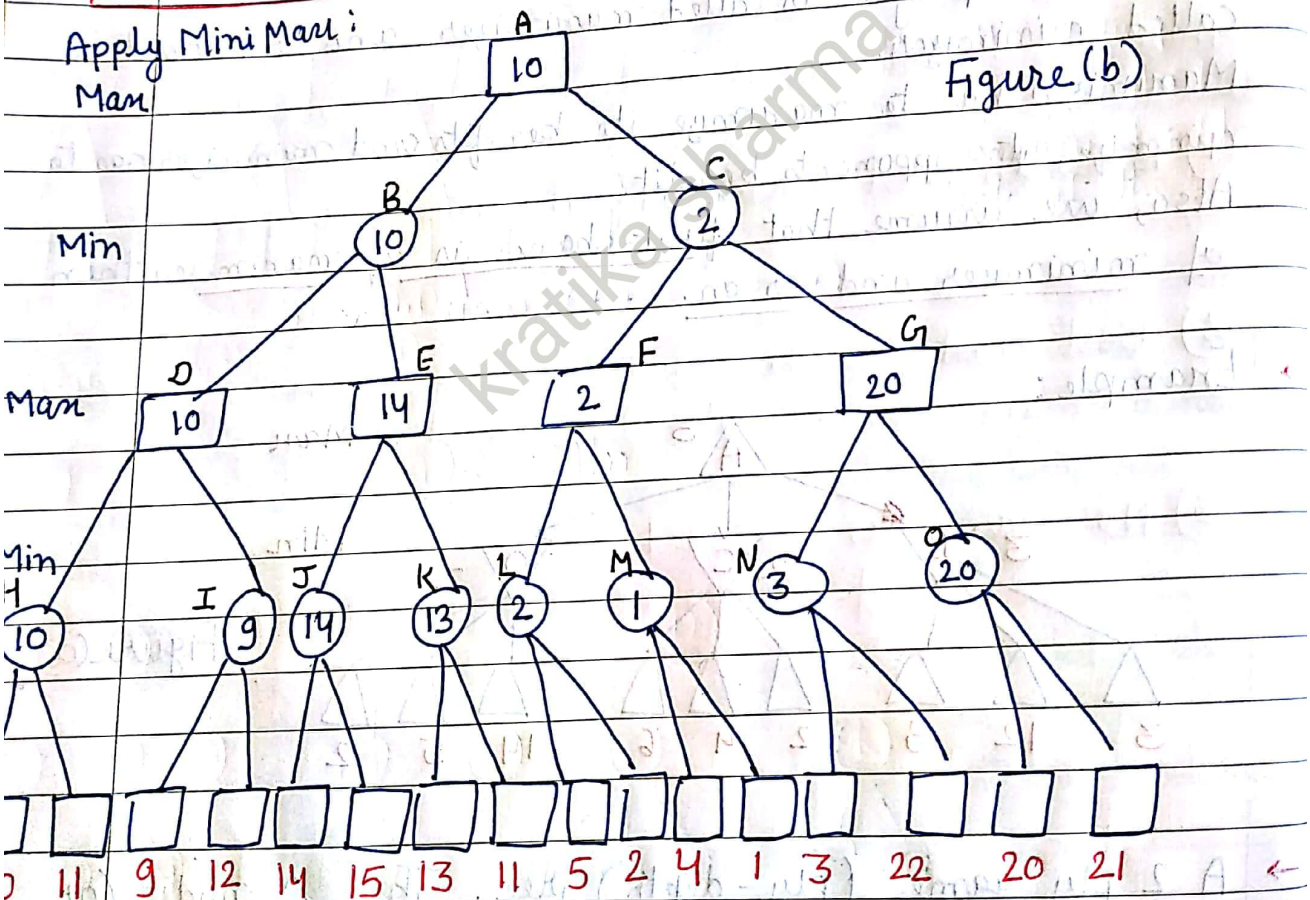


Figure (b)

Step 1: See bottom left nodes - 10, 11

So for the above min node value will be

min (∞, 10) = 10

min (10, 11) = 10

So we gat 10 at min node H.

So assign 10

In minimax strategy always travel level by level, starting from the terminal nodes. DATE

for node I

Step 2: min (-∞, 9) = 9

min (9, 12) = 9

I = 9.

for node K

Step 3: min (-∞, 14) = 14

min (14, 15) = 14.

like wise calculate value for all the Min nodes (L, M, N & O).

Step 4: Here now we have to assign values for max nodes

for node D

max (-∞, 10) = 10

max (10, 9) = 10

D = 10

Step 5: for node E

max (-∞, 14) = 14

max (14, 13) = 14

E = 14.

like wise we get F = 2 & G = 20

Step 6: Now again we have min nodes.

for node B

min (∞, 10) = 10

min (10, 14) = 10

B = 10

Step 7: for node C

min (∞, 2) = 2

min (2, 20) = 2

C = 2

Step 8: node A maximizers move

max (-∞, 10) = 10

max (10, 2) = 2

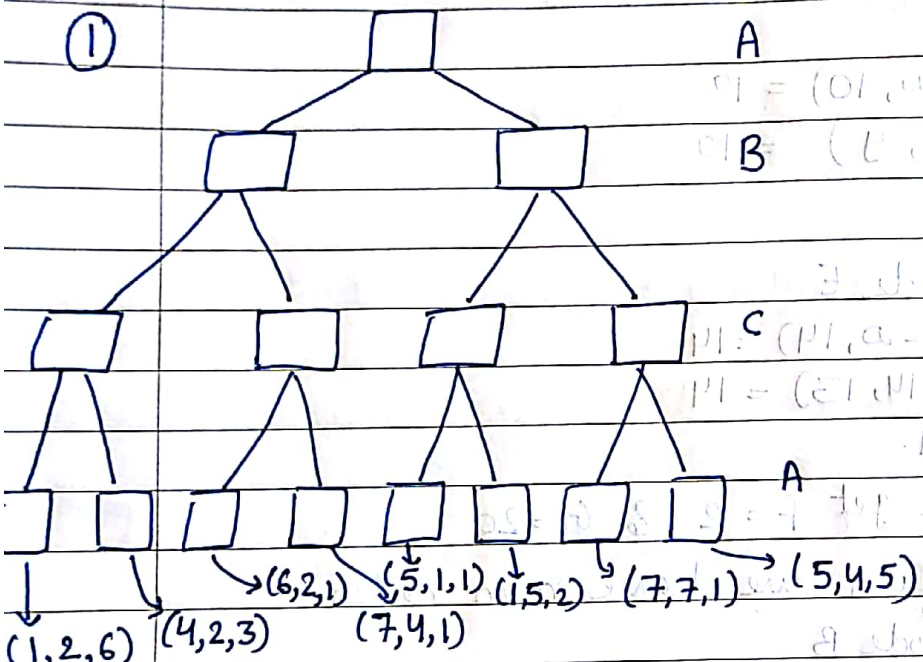
A = 10

→ The minimax algorithm performs a complete depth first exploration of the game tree. If the maximum depth of the tree is m and there are b legal moves at each point, then the time complexity of minimax is $O(b^m)$

Space complexity $O(bm)$

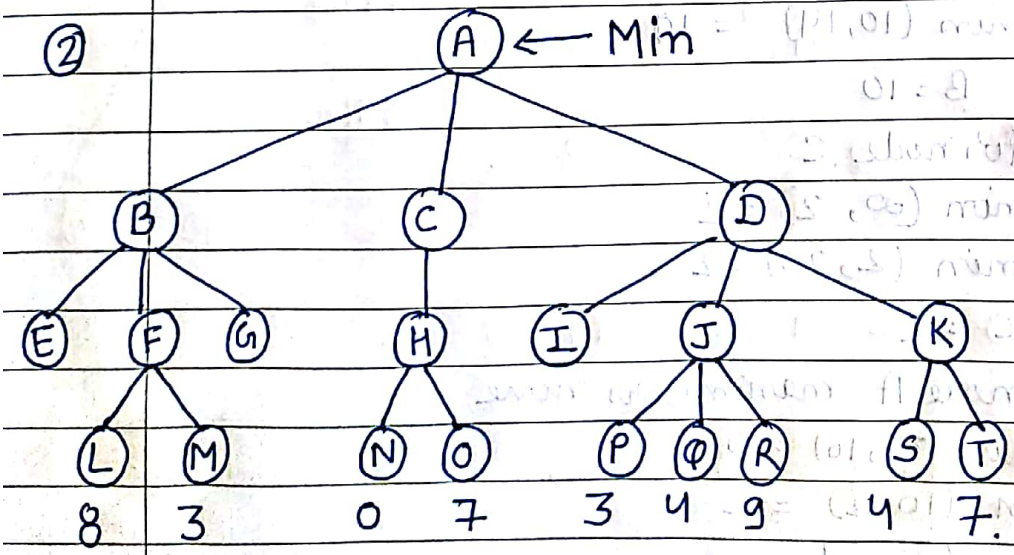
Perform minimax algorithm for the following trees:

① to move A



The first 3 plies of the game tree with 3 players (A, B, C) Each node you should mark with the values from the view point of each player. The best move you should mark at the root.

②



DATE

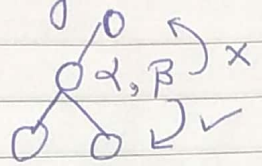
ALPHA - BETA CUT OFFs

→ α denotes max value
worst case = $-\infty$

→ β denotes min value
worst case = ∞

→ At max level we will change value of α
β value will remain constant
At min level we will change value of β
α value will remain constant.

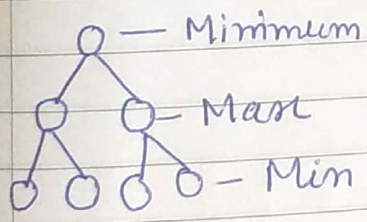
→ Never take any value upside the tree

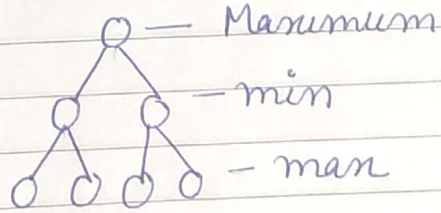


→ Don't forget to check the condition

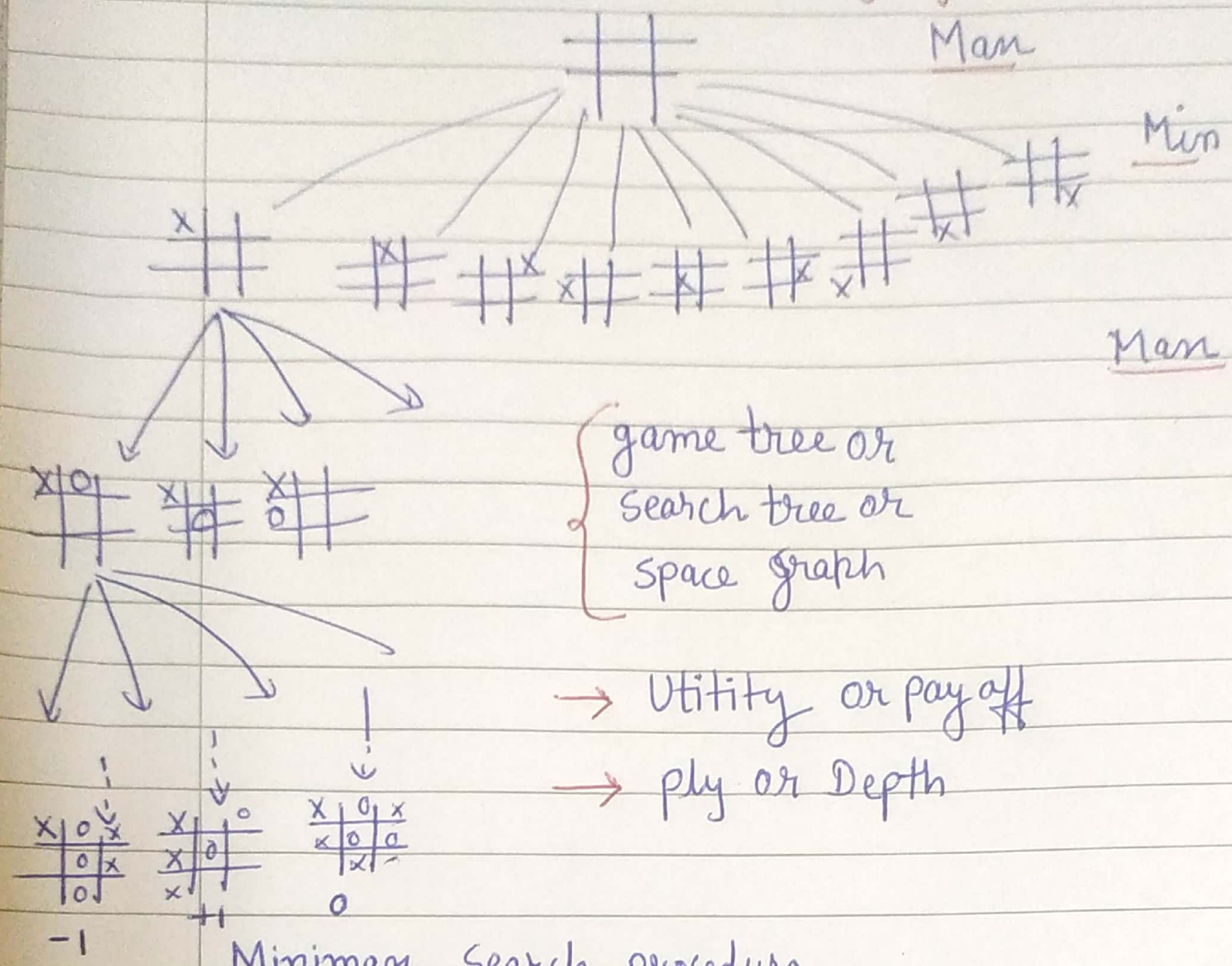
$$\alpha \geq \beta$$

→





Introduction to Game Playing



Minimax Search procedure

1. Generate whole tree to leaves
2. Apply evaluation function to the leaves
3. Backup values from leaves towards the goal
   - (a) Max node compute max value of its child nod
   - (b) Min node compute min value of its child node
4. When the value reaches to the root choose the max value and corresponding node.

α-β Cut off

Max Cut off : at a max node n, cut off search if $\alpha(n) >= \beta(n)$

Min Cut off : at a min node n, cut off search if $\beta(n) >= \alpha(n)$

Minimax Algorithm

Backtracking Algorithm
Best move strategy used
Max node will try to maximize its utility (Best Move)
Min will try to minimize utility (Worst Move)

Time Complexity $O(b^d)$

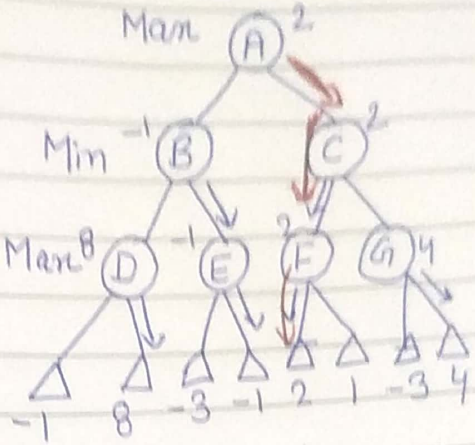
Eg: $= 3^2$

$= 9$

chess - 35 moves possible on an average

depth - mean avg 50

$(35)^{100}$

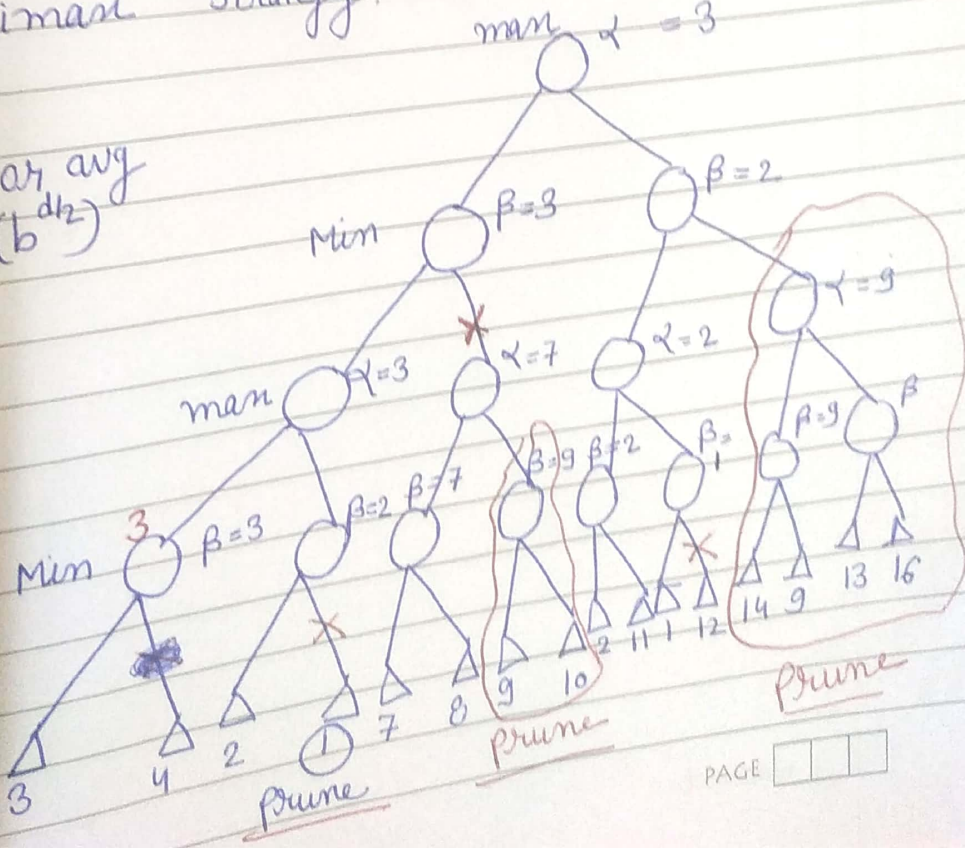


A → C → F → 2

Alpha Beta Pruning (α-β)

Cut off search by exploring less no. of nodes as compare to minimax strategy.

Best case or avg T.C = $O(b^{d/2})$



CONSTRAINT SATISFACTION

DATE

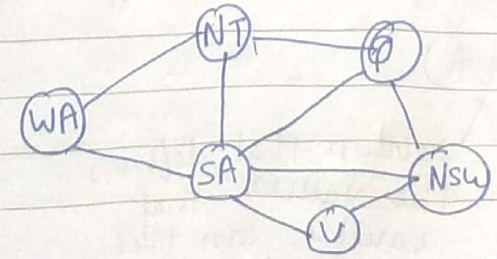
→ A CS. problem consists of 3 components, X, D and C:
X is a set of variables $\{X_1, \ldots X_n\}$
D is a set of domain $\{D_1, \ldots D_n\}$ one for each variable.
C is a set of constraints that allowable combinations of values.

→ Each domain D_i consists of a set of allowable value $\{V_1, \ldots V_K\}$ for variable X_i.
Each constraint C_i consists of a pair <scope, rel>, where scope is a tuple of variable that participate in the constraint and rel is a relation that defines the values that those variables can take on.

→ Example Problem: Map coloring
We are given the task of coloring each region either red, green or blue in such a way that no neighboring regions have the same color.

Tasmania

Constraint graph:



DATE

To formulate this as a CSP, we define the variables to the regions

$X = \{WA, NT, Q, NSW, V, SA, T\}$

$D_i = \{red, green, blue\}$

$C = \{SA \neq WA, SA \neq NT, SA \neq Q, SA \neq NSW, SA \neq V, WA \neq NT, NT \neq Q, Q \neq NSW, NSW \neq V\}$

	WA	NT	SA	Q	NSW	V	T
Initial Domain	RGB	RGB	RGB	RGB	RGB	RGB	RGB
After WA = R	R	GB	GB	RGB	RGB	RGB	RGB
After NT = G	R	G	B	R	G	R	RGB
After T = R	R	G	B	R	G	R	R

Unique values are assigned to states, satisfied with domain value.

CSP Consists of 3 Components

V set of variables

D set of Domains

C set of Constraints that specify allowable combination of value

$C_i = (scope, rel)$

scope: Set of variables that participate in Constraint

rel: relation that defines the values that variable can take

Example Problem: Cryptarithmatic Puzzle

Variables : letters {A-Z} and carry digits
Domains : digits {0-9}
Constraints : Arithmatic constraints and all letters (A-Z) distinct.

Example ① SEND + MORE = MONEY ② BASE + BALL = GAMES

Assign decimal digit to each of the letters in such a way that the answer to the problem is correct to the same letter occurs more than once, it must be assign the same digit each time. no two different letters may be arrigned the same digit.

① $C_4\ C_3\ C_2\ C_1$
S E N D
M O R E
M O N E Y

② $C_3\ C_2\ C_1$ ← carries
B A S E
B A L L
G A M E S
1

③ T H I S
I S
H E R E

④ T W O
T W O
F O U R

⑤ R O A D S
C R O S S
D A N G E R

⑥ D O N A L D
G E R A L D
R O B E R T

⑦ L O G I C
L O G I C
P R O L O G

② G → 0 or 1 ✓

E + L = S
E + L = S + 10 (carry)
E = S − L + 10

Constraint equations are

E + L = S → C1
S + L + C1 = E → C2
2A + C2 = M → C3
2B + C3 = A → C4
G = C4

G has to be non-zero digit, so the value of carry C4 should be 1 and hence G = 1

1. $G_1 = C_4 \Rightarrow \boxed{G = 1}$

2. $2B + C_3 = A \rightarrow C_4$

2.1 $C_4 = 1$, therefore $2B + C_3 > 9$ means B can take values from 5 to 9.

2.2 • if $B = 5$ → if $C_3 = 0 \Rightarrow A = 0 \Rightarrow M = 0$ for $C_2 = 0$ or $M = 1$ for $C_2 = 1$ ✓

→ if $C_3 = 1 \Rightarrow A = 1$ ✗ (as G is already 1)

• if $B = 6$ we get similar contradiction while generating the search tree

• if $\boxed{B = 7}$ then for $C_3 = 0$, we get $\boxed{A = 4} \Rightarrow M = 8$ if $C_2 = 0$ that leads to contradiction later, so this path is pruned.

if $C_2 = 1$, then $\boxed{M = 9}$

3. let us solve $S + L + C_1 = E$ and $E + L = S$

→ $2L + C_1 = 0$

$\boxed{L = 5}$ and $C_1 = 0$

$[E + L + L + C_1 = E$, $2L + C_1 = 0]$

→ using $\boxed{L = 5}$, we get $S + 5 = E$ that should generate carry $C_2 = 1$

→ So $S + 5 > 9 \Rightarrow$ possible values for E are $\{2, 3, 6, 8\}$ with carry bit $C_2 = 1$

if $E = 2$, then $S + 5 = 12 \Rightarrow S = 7$ ✗

if $E = 3$, then $S + 5 = 13 \Rightarrow S = 8$

therefore $\boxed{E = 3}$ and $\boxed{S = 8}$ are fixed up.

$C_4\ C_3\ C_2\ C_1$

$$\begin{array}{r} SEND \\ ① \quad MORE \\ \hline MONEY \end{array}$$

Constraint Equations

$D + E = Y \longrightarrow C_1$

$N + R + C_1 = E \rightarrow C_2$

$E + O + C_2 = N \rightarrow C_3$

$S + M + C_3 = O \longrightarrow C_4$

$M = C_4$

$M = C_4 \Rightarrow \boxed{M = 1}$

$S + M + C_3 = 0 \quad C_3 \begin{cases} 0 & S + 1 + 0 = 0 \quad S + 1 = 0 \\ 1 & S + 1 + 1 = 0 \quad S + 2 = 0 \end{cases}$

So S can be either 8 or 9

$\boxed{S = 9}$ $\boxed{O = 0}$