Applications of AI:-

(1) **AI in Gaming**: Chess, Poker, tic-toe.
↳ machine can think large no. of moves.

(2) **AI in NLP**: Natural lang. Processing
↳ Machine can understand human lang.

(3) **AI in HealthCare**: Fast diagonsis
↳ Robotic Surgery.

(4) **AI in Finance**: Adaptive Intelligence.
↳ automatic chatbots, algorithm trading.

(5) **AI in Data Security**: Helps in making data/appln more secure.
↳ AEG bot, AI2

(6) **Expert System**: Integration of s/w, machine and special infon to provide reasoning & advise.

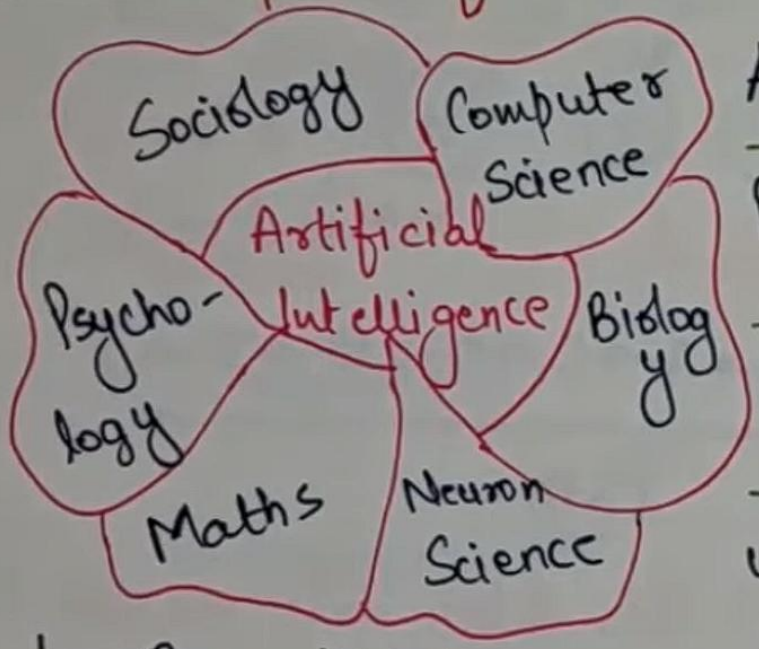
(7) **Computer Vision**: Understand the visual automatically by machine.

(8) **Speech Recognition**: Extract the meaning of sentence by human talk. [slang removal, noise rem.]

(9) **Robotics**:
↳ Erica and Sophia.
Talk and behave like humans.

(10) **AI in e-Commerce**: Automatic recommend

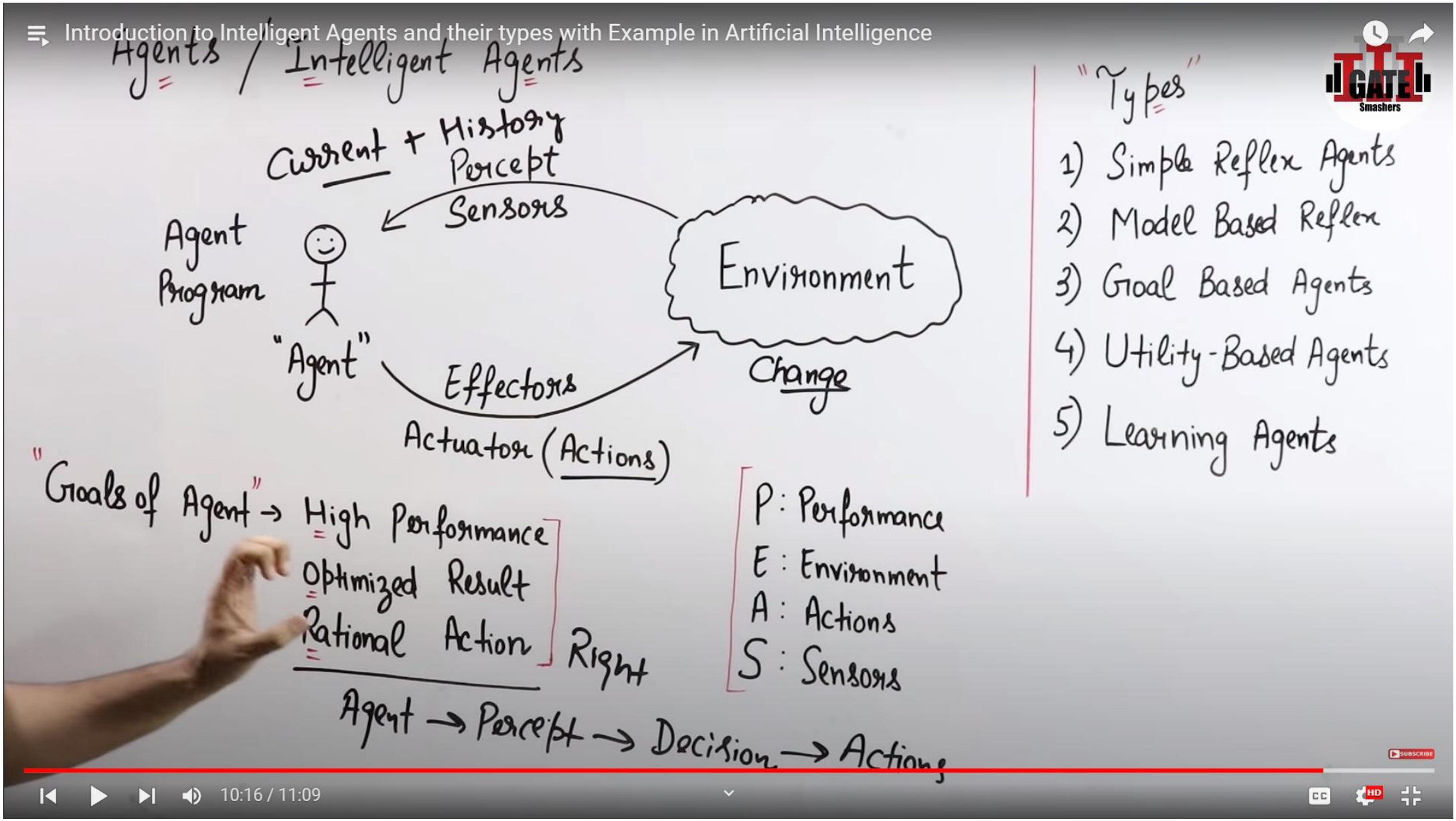
AI is Comprised of :-



- Reasoning
- Learning
- Problem solving
- Language Understanding.

Advantages of AI	Disadvantages of AI
Accuracy ↑ & Error ↓	COST ↑
Fast Decision Making.	Can't thing beyond the limits.
Reliability is more	No feeling & emtions.
usefulness in Risky Area.	more dependency on machines ↑.
Digital Assistant	No original thinking.

Introduction to Intelligent Agents and their types with Example in Artificial Intelligence



10:16 / 11:09

Types of AI Agents:-

[Responsible for any work output obtained from system.

AI is defined as Study of Rational Agents.

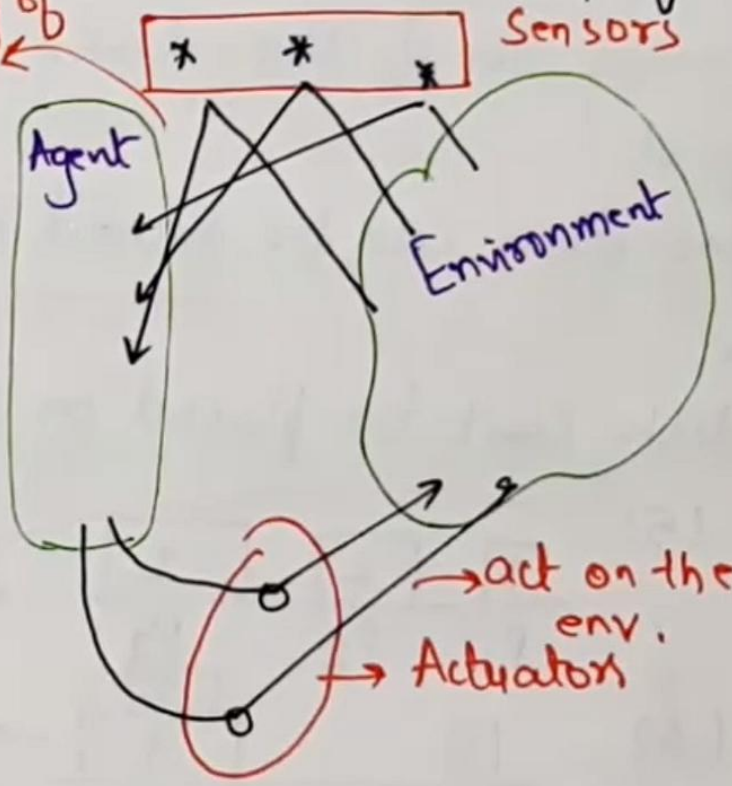
- Person
- Firm
- Machine/s/w

Make Decisions

AI System is Composed of
- Agent
- Environment

Agent → Info^n gathering.
→ action perform.

Perceiving of Info^n.

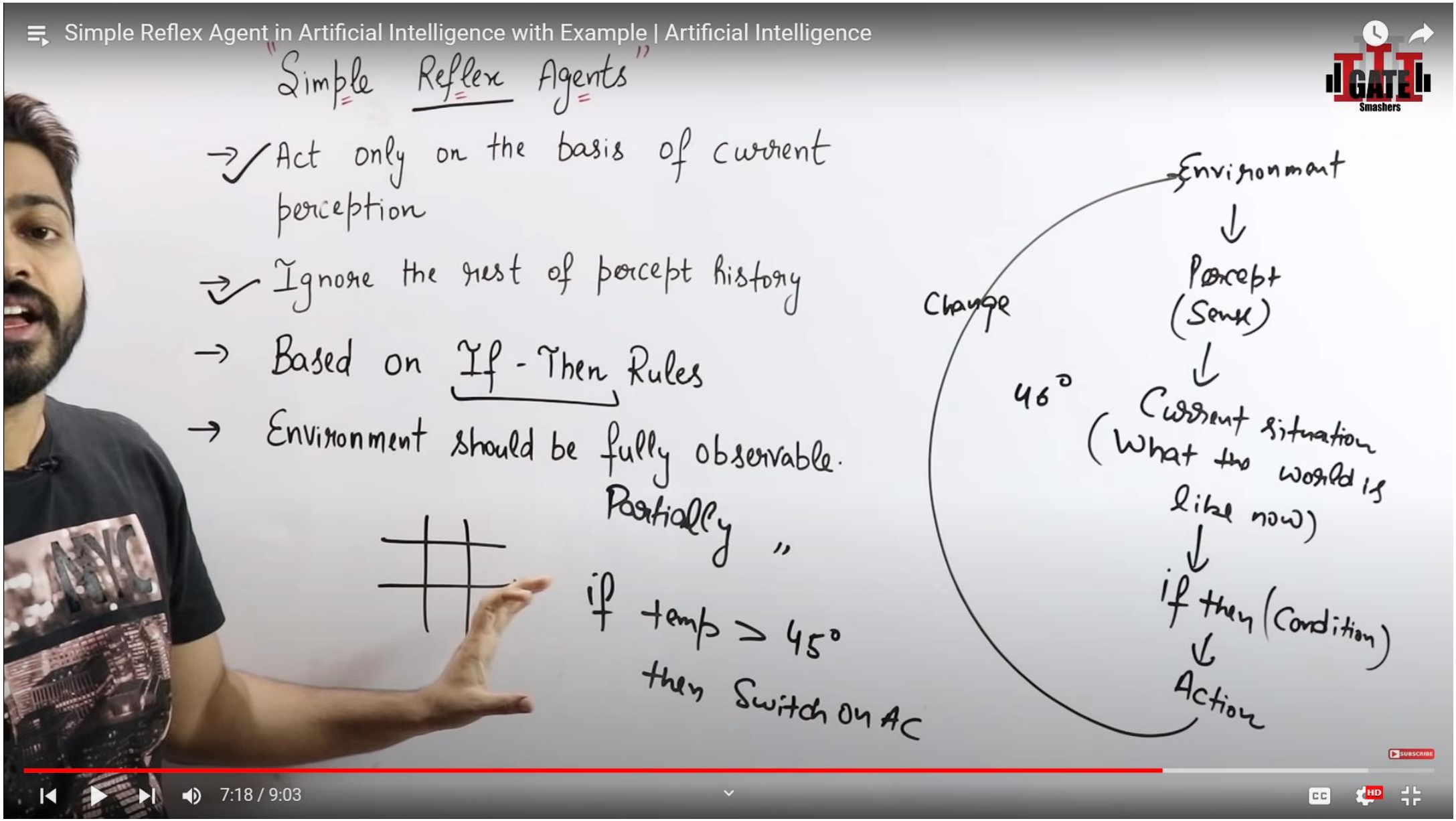


Agent is anything like:-

i) perceiving its environment through **Sensors.**

ii) Acting upon that environment through **actuators.**

Simple Reflex Agent in Artificial Intelligence with Example | Artificial Intelligence

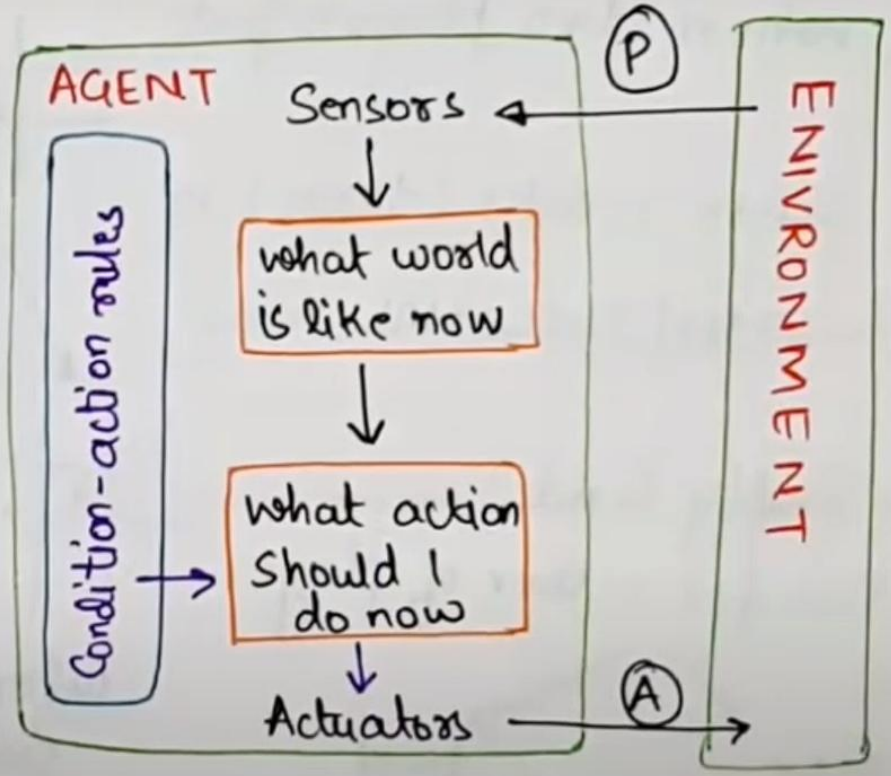


7:18 / 9:03

Simple Reflex Agents:

Works only on Current situation/ perception and ignores the history of previous State. True

↳ Condition-Action Rule



Limitations:-

↳ i) Very limited Intelligence.

ii) No Knowledge about non-perceptual parts of State

iii) Can go into infinite loop.

(2) Model based Reflex Agent:

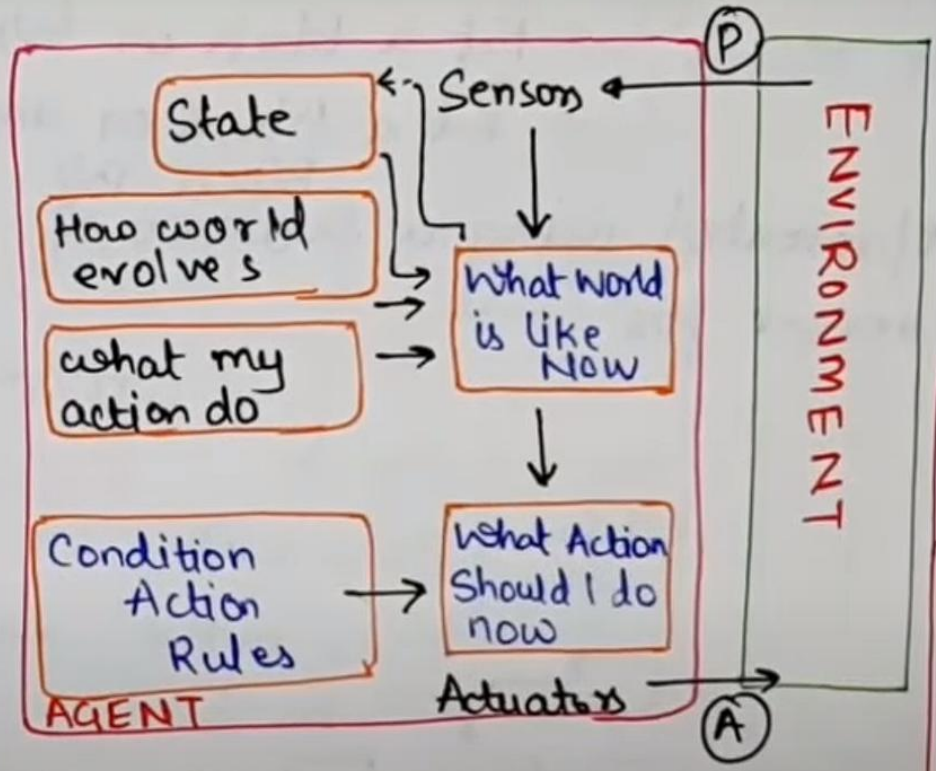
↳ Works by finding the rule whose Cond^n matches Current Situation.

↳ Can work in partially observable env., and track situation.

↳ Agent keeps track of internal State which is adjusted by each percept and that depends on percept history.

↳ Model: How things happen in world.

↳ Agent State updation required Info^n:
- (1) How world is evolving
- (2) How agent's action affect the word.



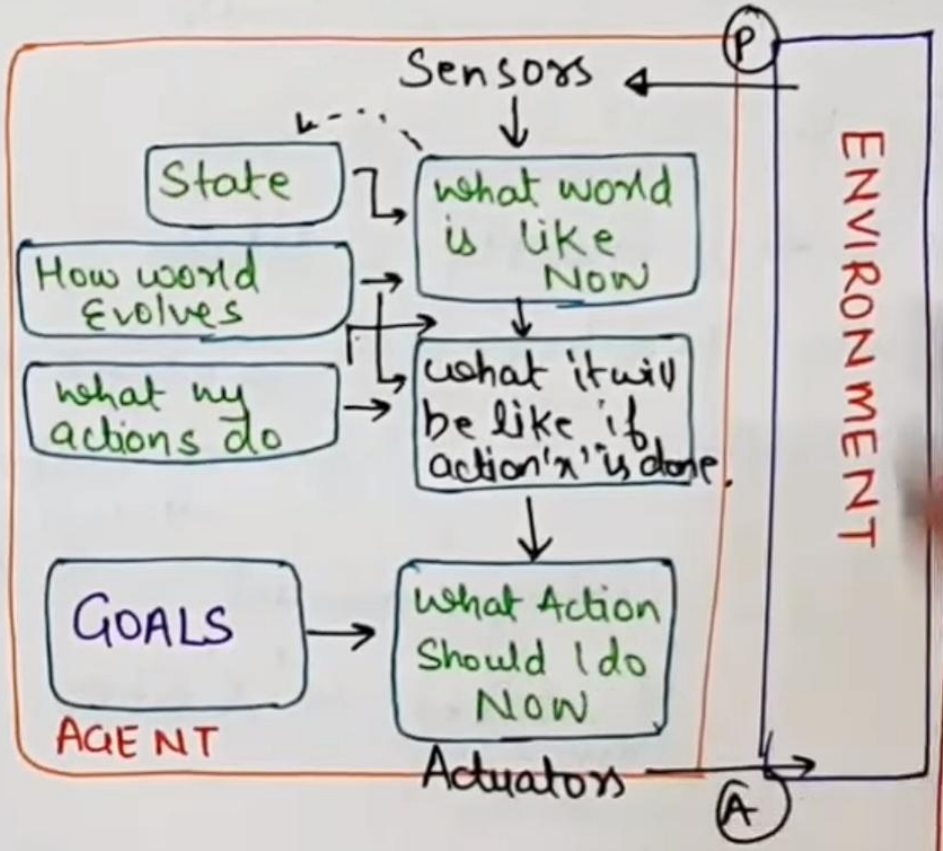
Goal-Based Agents:

→ Focuses only on reaching the goal set.

↳ Agent takes decision based on how far it is currently from the Goal State.

↳ Every action is taken to minimize distance to Goal State.

↳ More flexible Agent.



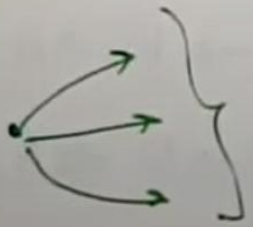
L6: Artificial Intelligence Agents | Types of AI Agents | Artificial Intelligence Lectures in Hindi

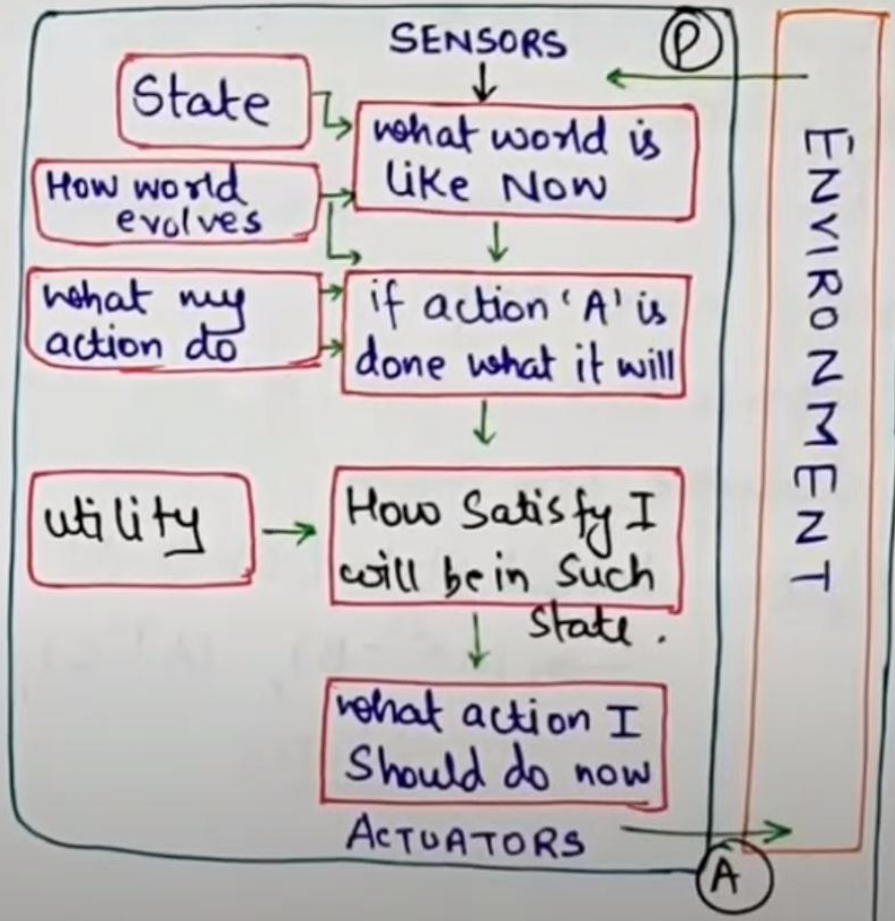
3) Utility-Based Agents:-

↳ Agents are more Concerned about the utility (Preference) for each state.

↳ Act based not only on goals but also the best way to achieve goal.

↳ Useful when there are multiple possible alternatives and agent has to choose in order to perform best action.

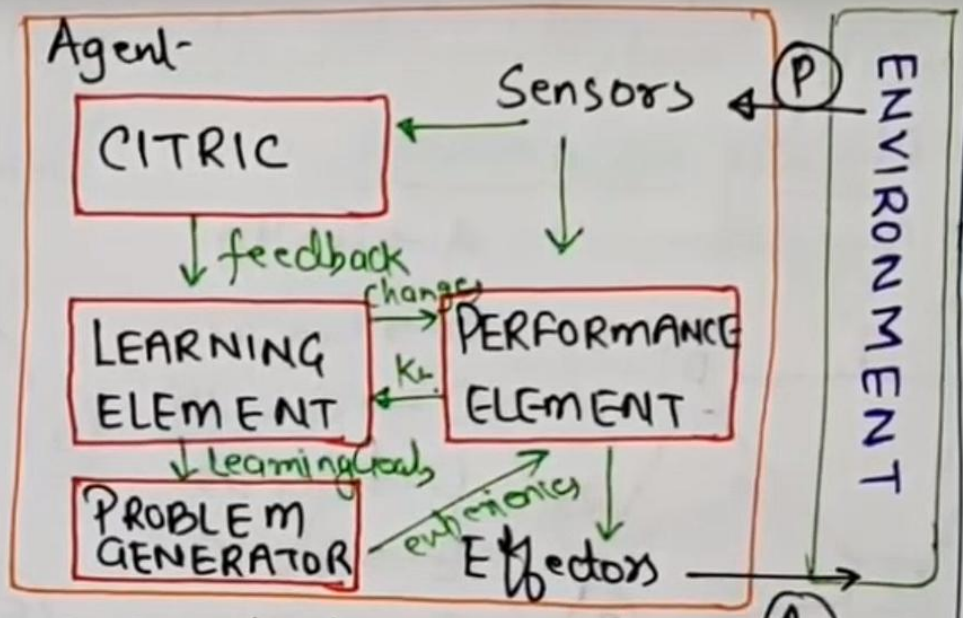




5) Learning Agents:-

↳ Can Learn from its past experiences.

↳ Starts to ACT with basic knowledge and then able to act by adapting Learning.



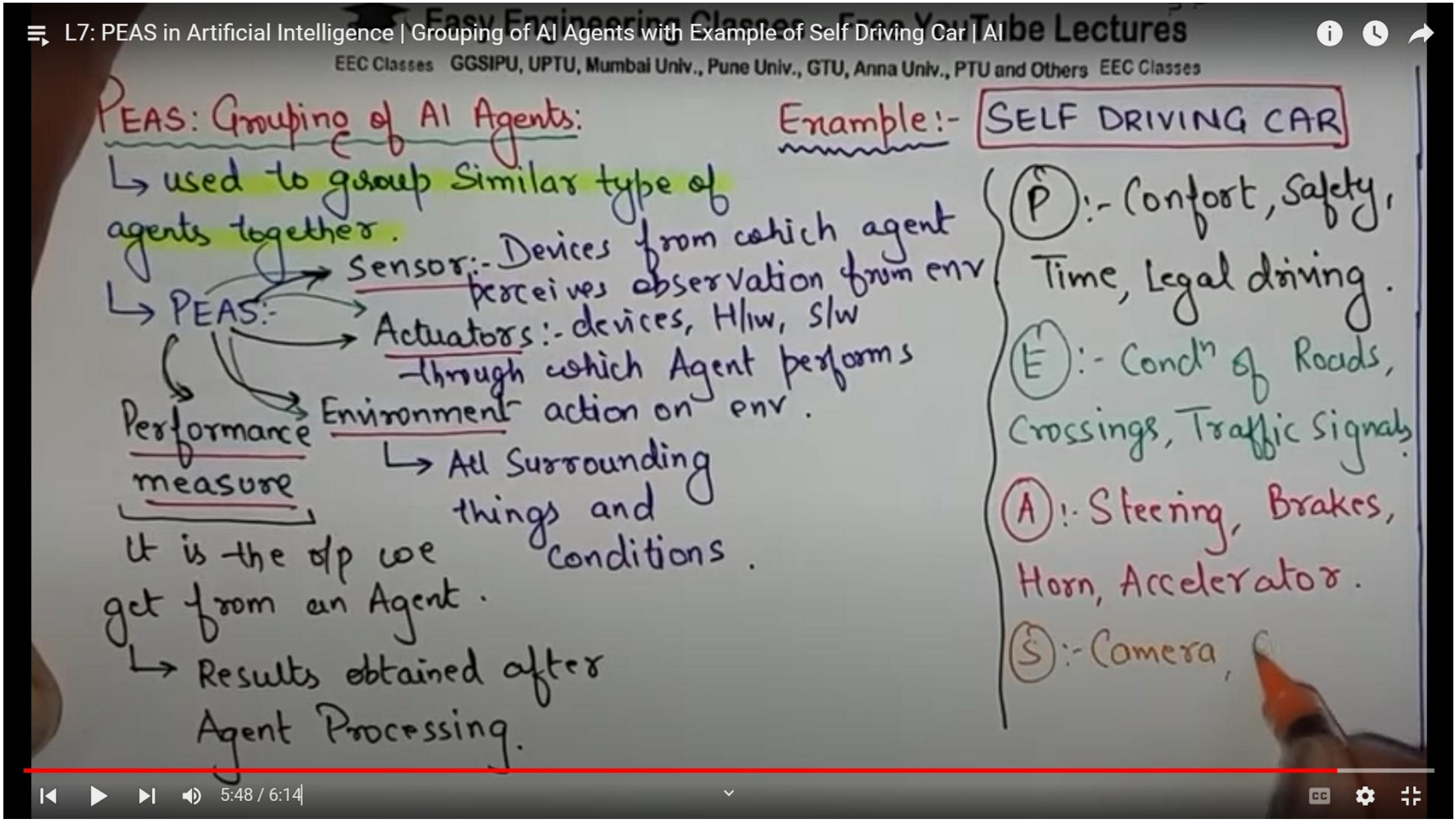
Components:-

(i) Learning Element: Makes Improvement in System by learning from env.

(ii) Critic: Gives feedback about Agent's Performance based on Standard.

(iii) Performance Element:

(iv) Problem Generator:


L7: PEAS in Artificial Intelligence | Grouping of AI Agents with Example of Self Driving Car | AI
Easy Engineering Classes - Free YouTube Lectures
EEC Classes GGSIPU, UPTU, Mumbai Univ., Pune Univ., GTU, Anna Univ., PTU and Others EEC Classes
PEAS: Grouping of AI Agents:
↳ used to group Similar type of agents together.
↳ PEAS:-
Sensor:- Devices from which agent perceives observation from env
Actuators:- devices, H/w, S/w through which Agent performs action on env.
Environment
↳ All Surrounding things and conditions.
Performance measure
It is the o/p we get from an Agent.
↳ Results obtained after Agent Processing.
Example:- SELF DRIVING CAR
(P):- Confort, Safety, Time, Legal driving.
(E):- Condn of Roads, Crossings, Traffic Signals
(A):- Steering, Brakes, Horn, Accelerator.
(S):- Camera,
5:48 / 6:14

problem solving | Artificial intelligence | Lec-9 | Bhanu Priya

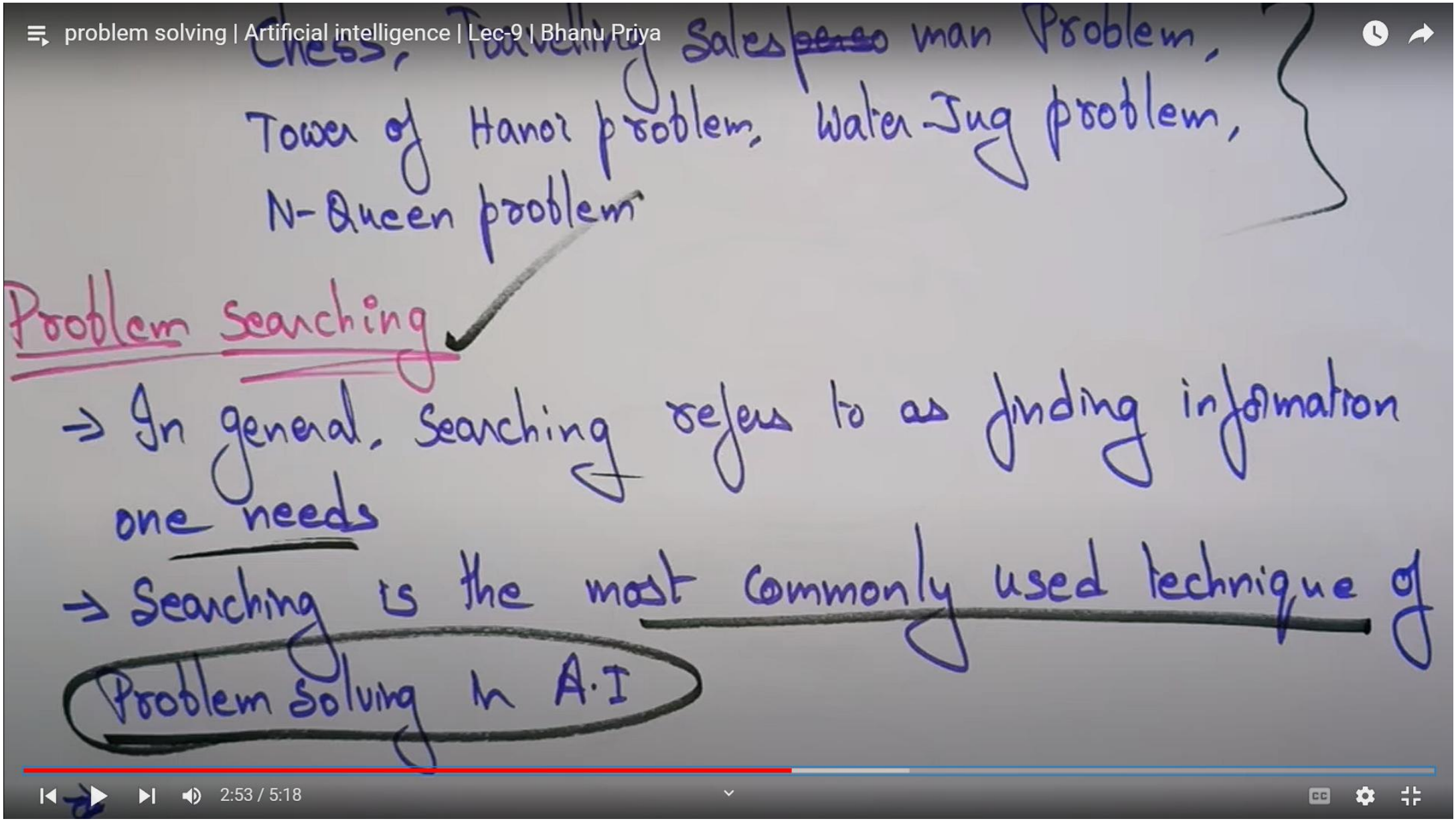
Problem Solving

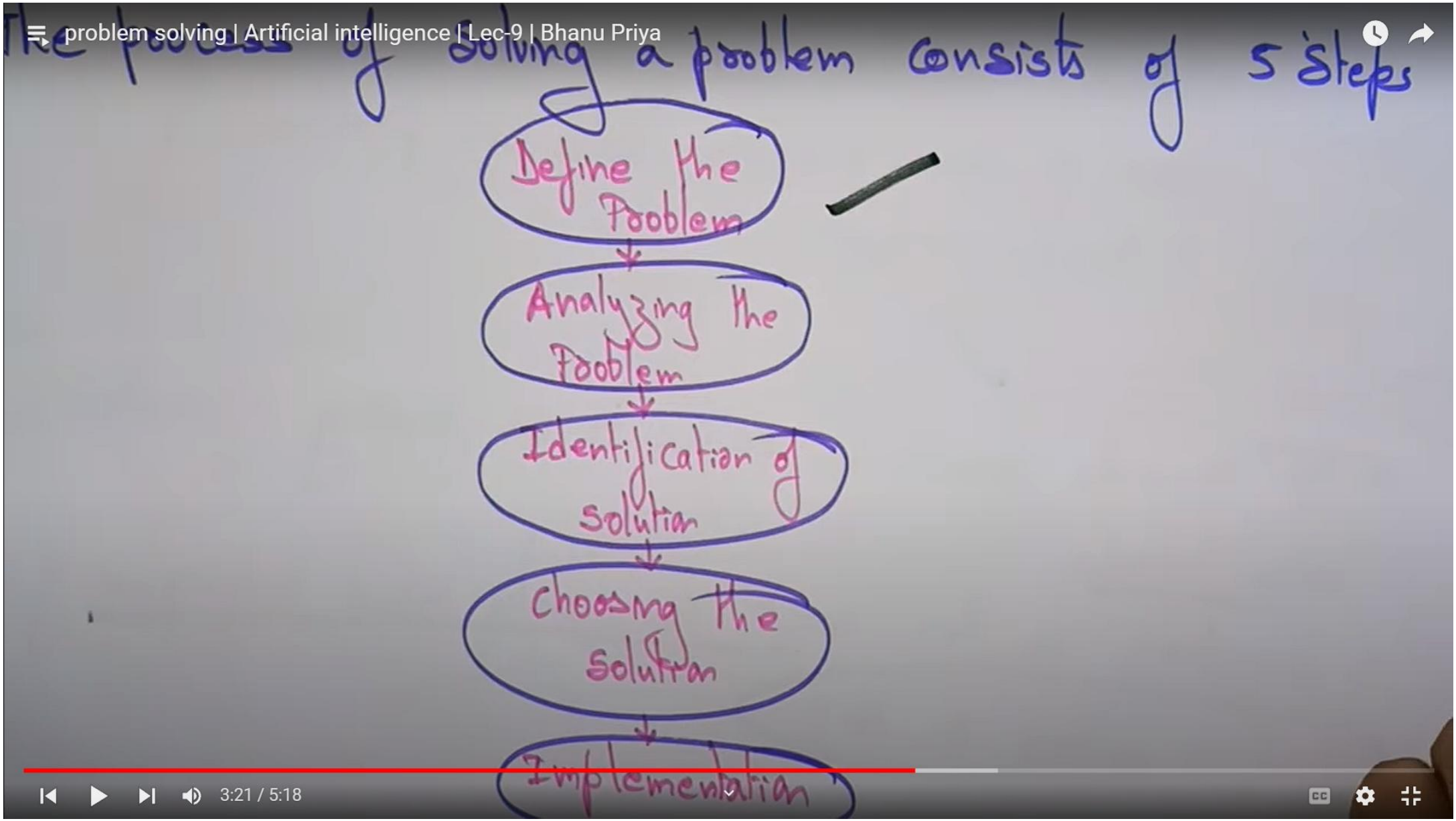
* Problem Solving is in games such as Sudoku eg. It c be done by building an A.I system to solve that particu Problem.

* To do, this one needs to define the problem stateme first and then generating the solution by keeping the Condition in mind.

* Some of the most popularly used problem solving with the help of AI are:

0:55 / 5:18

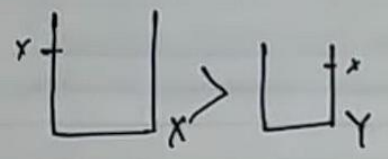

problem solving | Artificial intelligence | Lec-9 | Bhanu Priya
Chess, Travelling Salesman Problem,
Tower of Hanoi problem, Water Jug problem,
N-Queen problem
Problem Searching
⇒ In general, searching refers to as finding information one needs
⇒ Searching is the most commonly used technique of Problem Solving in A.I
2:53 / 5:18


problem solving | Artificial intelligence | Lec-9 | Bhanu Priya
The process of solving a problem consists of 5 steps
Define the Problem
Analyzing the Problem
Identification of solution
Choosing the solution
Implementation
3:21 / 5:18

WATER JUG Problem and its State Representation:

Two Jugs of different Capacities are given.

X-litre Y-litre



No Marking is there on any JUG.

↳ Goal is to fill enactly 'L' litres of water into 'Y'-litre JUG.

State is represented as <x, y>

- x → integer amount of water in 'X' litre JUG
- y → integer amount of water in 'Y' litre JUG

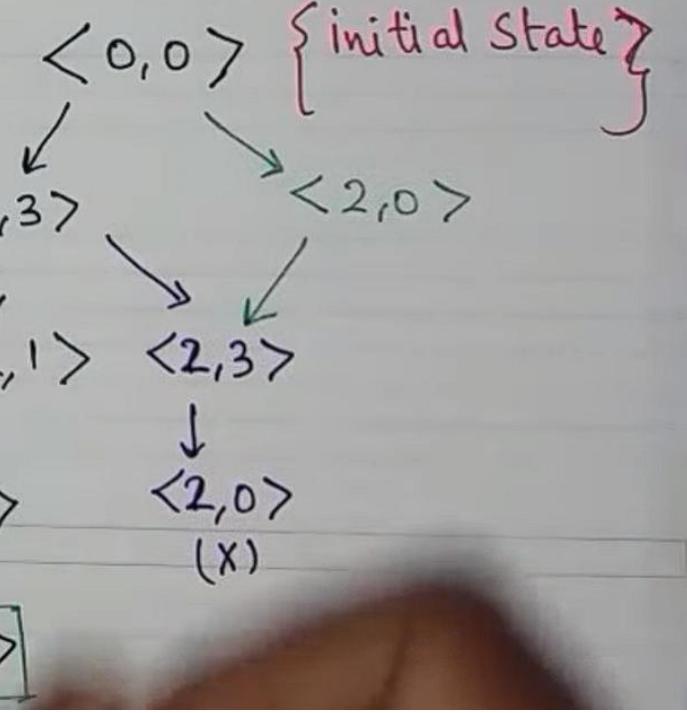
<1,2>

X Y

Example:- Suppose Capacity of two jugs → 2 litre, 3 litre

<x, y>

GOAL: To Get enactly '1' liter of water in 2 liter JUG. <1,0> (Goal State)



L17: Breadth First Search(BFS) in Artificial Intelligence with Solved Examples | Uninformed Search

Breadth First Search:

↳ Explores all the nodes at given depth before proceeding to the next Level.

↳ Uses Queue to implement.

ALGORITHM:

i) Enter Starting nodes on Queue.

ii) If Queue is empty, then return fail and stop.

iii) If first element on Queue is GOAL Node, then return Success and Stop.

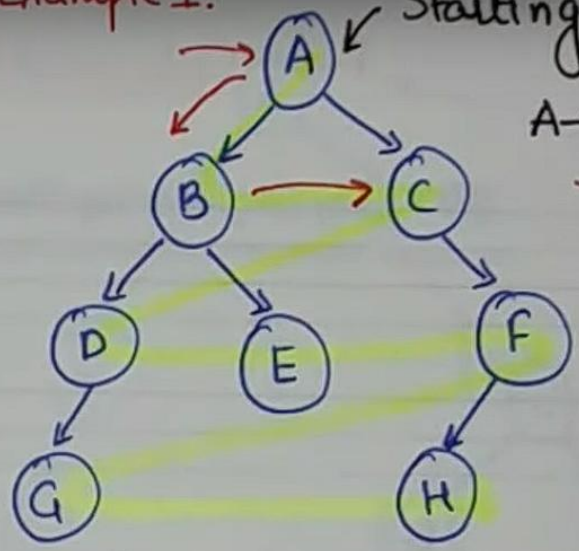
ELSE

(IMP)

iv) Remove and expand first element from Queue and place children at end of Queue.

v) Goto Step (ii)

Example 1:-



A → B → C → D → E
↓
F
↓
G
↓
H

Initial Queue {A} = Goal node x

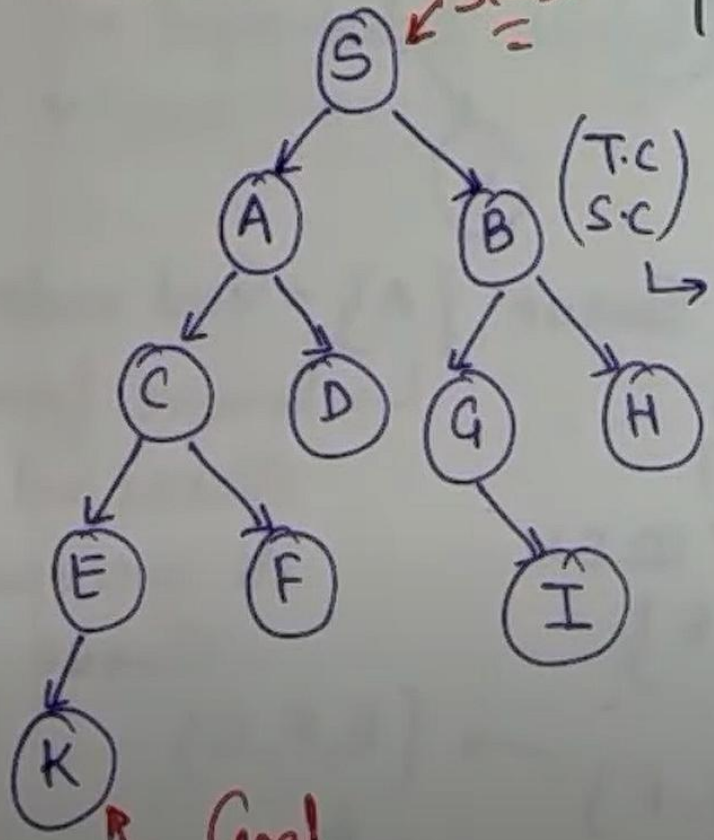
↳ Remove from Queue and add its successor to Queue.

{B, C}
↓
(D, E)
{C, D, E}
↓
{D, E, F} → {E, F, G}
↓
{G, H}

6:01 / 10:57

L17: Breadth First Search(BFS) in Artificial Intelligence with Solved Examples | Uninformed Search

Example 2:-



(T.C
S.C) → $O(b^d)$
↳ branching factor

{S}
↳ {A, B}
↳ {B, C, D}
↳ {C, D, G, H}
↳ {D, G, H, E, F}
↳ {H, E, F, I}
↳ {F, I, K}

S → A → B → C → D → G → H → E → F → I → K

Adv:- find a Soln if its exists.
↳ minimal Soln in Least no. of steps.

Disadv:- More memory
↳ need lots of time
↳ Soln is far from root node.

Artificial Intelligence:- AI is the study of How to make computer do things which people do better. [machine + human Intelligence]

↳ AI Can Cause a machine to work as human.

↳ AI → Artificial [Man-Made]
→ Intelligence [Power of thinking]

GOALS OF AI: i) Replication of Human Intelligence.

ii) Solving problems - that require Knowledge.

iii) Building a machine that Can do human Intelligence task. [CHESS, Proving theorem, automated car driving ...]

iv) Providing advise to the user.

v) Intelligent Conn^n b/w a perception and

Reasons of Boost in AI:

↳ i) S/w or device Can be made to solve Real-time Problems.

ii) Creation of Virtual assistant [SIRI, CORTANA]

iii) Robots development. [Helps in dangerous env. cond^n]

iv) New JOB opportunities.

DFS: Depth-first Search.

↳ Recursive algorithm.

↳ Starts from root node and follows each path to its greatest depth node before moving to the next path.

↳ Implemented using STACK. (LIFO)

ALGORITAM: → (PUSH)

i) Enter ROOT node on Stack

ii) Do until Stack is not empty

↳ a) Remove Node → (POP)

↳ i) If Node = Goal Stop.

ii) Push all children of node in Stack

Adv:- Less memory

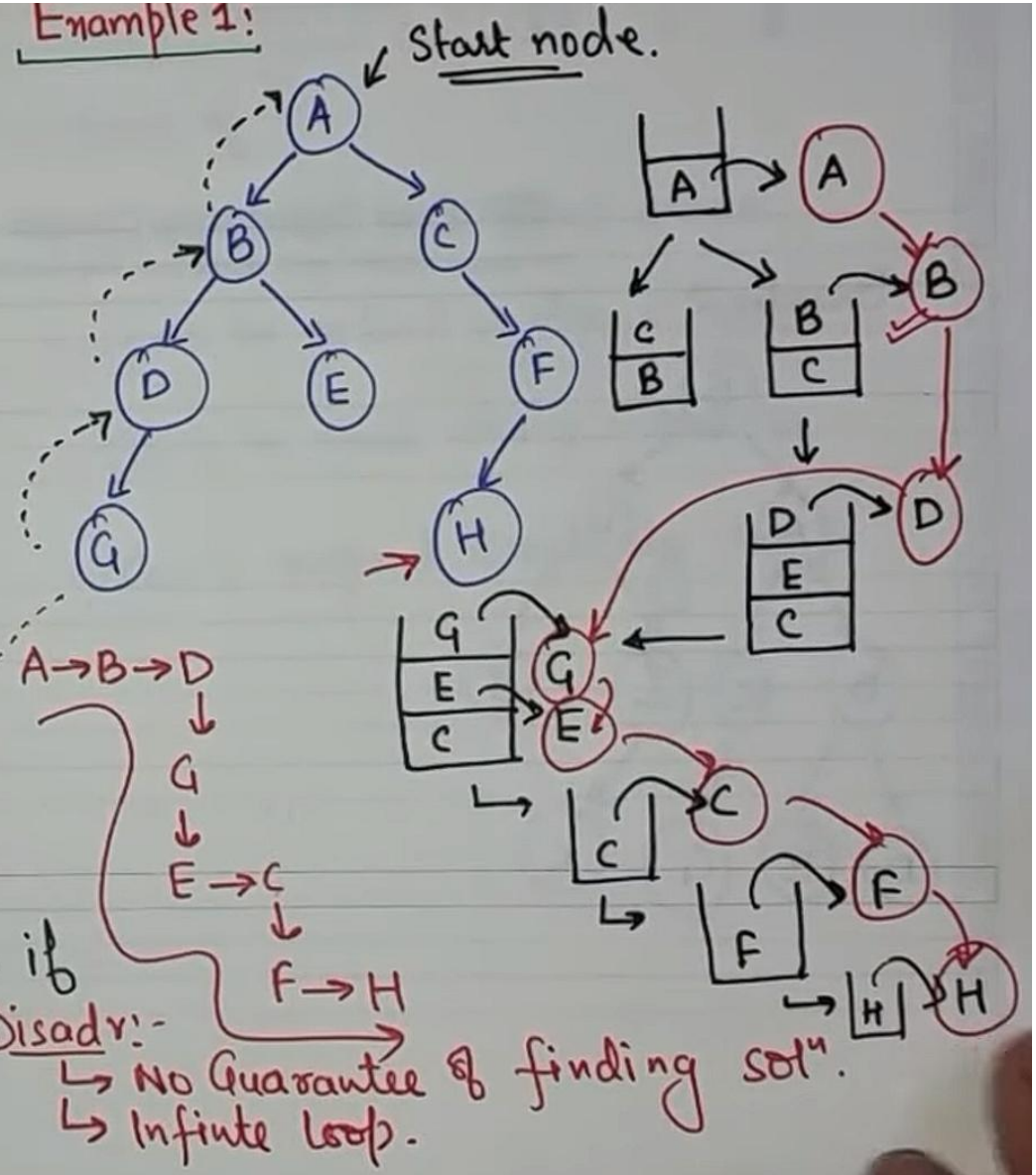
↳ Less time to reach goal node if traversal in right path.

Disadv:-

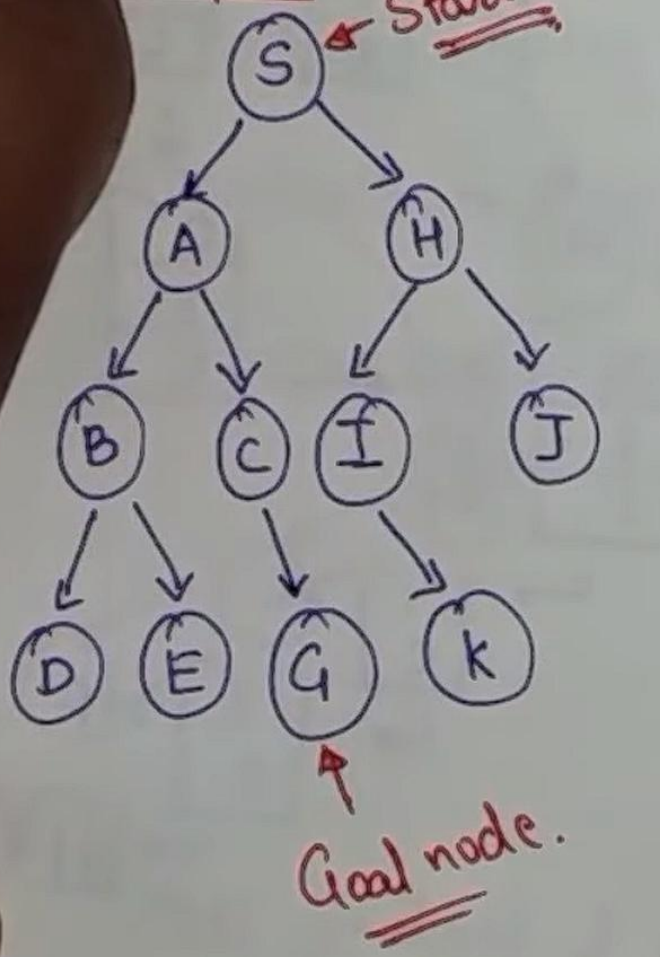
↳ No Guarantee of finding sol^{n}.

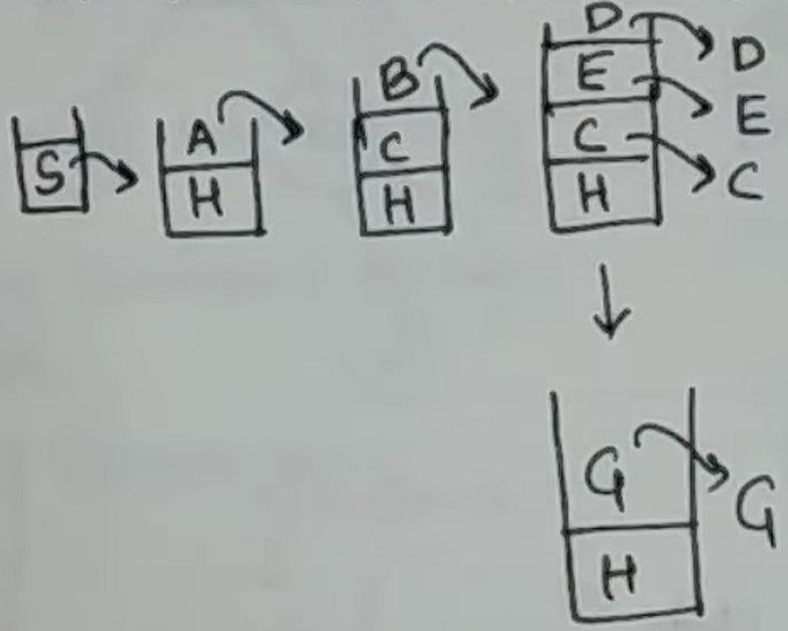
↳ Infinte loop.

Example 1:



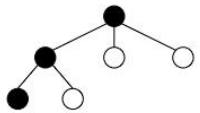
manuple 2:-





S → A → B → D → E → C → G

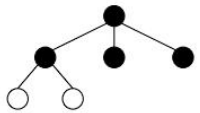
Space		
Optimal?		
Complete?		

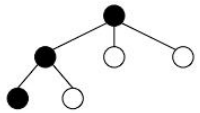


b: branching factor d: solution depth m: maximum depth

Search Strategies: Blind Search

Criterion	Breadth-First	Depth-First
Time	b^d	b^m
Space	b^d	b^m
Optimal?	Yes	No
Complete?	Yes	No





b: branching factor d: solution depth m: maximum depth

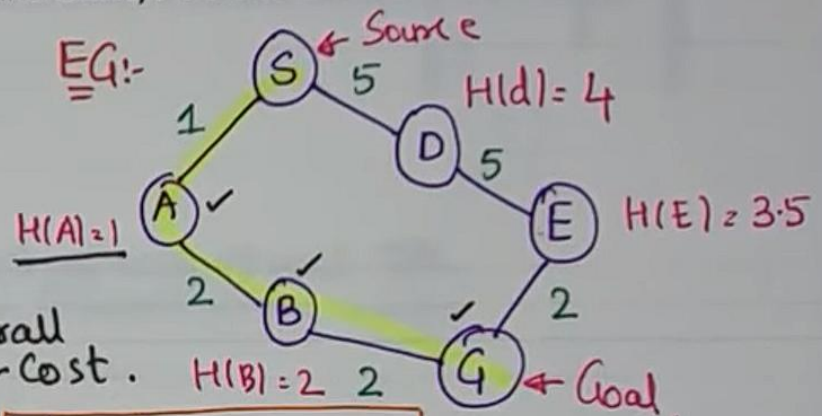
Informed Search:

- Heuristic Search
- Heuristic Function.

↳ Information about GOAL State is present.
↳ Better-than UnInformed Search.
↳ finds optimal Sol^{n} to reach GOAL State.
↳ Using HEURISTIC FUNCTION
- min. Path Cost.

↳ It is a Search which tries to reduce amount of Search that must be done by making intelligent choices for the nodes that are selected for expansion.

↳ Eg:-
- i) Best fit Search ALGO
- ii) A* Search ALGO.



$f(n) = G(n) + H(n)$ → for every node.

Path Cost — Heuristic Value

(A) $f(n)$ (1) + (1) = (2)

S → A (1) $f(n) = 1+1 = 2$
S → D (5) $f(n) = 5+4 = 9$
2 < 9

A → B $[f(n) = 3+2 = 5]$

S → A → B → G

L24: Heuristic Search in Artificial Intelligence | Admissible & Non-Admissible with Examples

Easy Engineering Classes Free YouTube Lectures

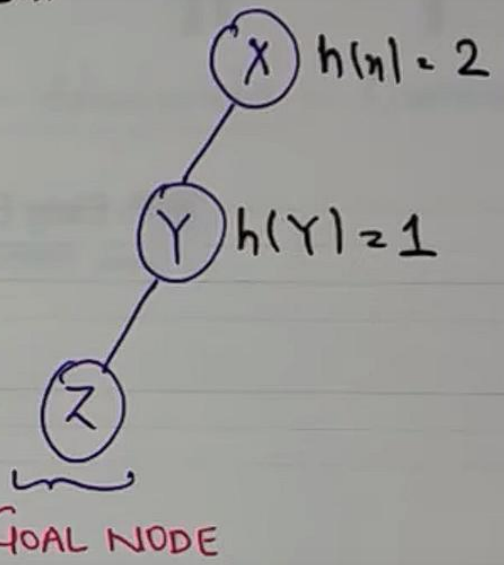
EEC Classes GGSIPU, UPTU, Mumbai Univ., Pune Univ., GTU, Anna Univ., PTU and Others EEC Classes

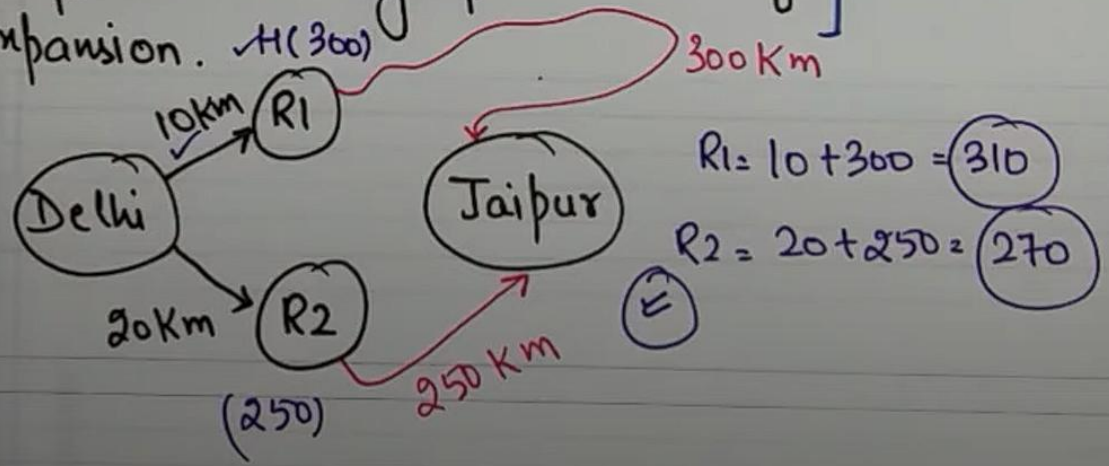
Heuristic Search:- → Tries to Solve Problem in minimum steps/cost.

↳ Tries to optimize a problem using heuristic function. [Informed Search]

Heuristic Function: It is a function H(n), that gives an estimation on the cost of getting from node 'n' to the GOAL State. } estimated Value.

[Helps in Selecting optimal node for expansion.





R1 = 10 + 300 = 310

R2 = 20 + 250 = 270

<=

5:07 / 10:21

Types of Heuristic:- (underestimate)

↳ (i) **Admissible:** In this Heuristic function, never overestimates the cost of reaching the Goal. $H(n) \leq H'(n)$ {Goal}

↳ h(n) is always less than or equal to actual cost of lowest-cost path from node 'n' to goal.

$H(B)=3$, $H(C)=4$, $H(D)=5$

$f(n) = H(n) + G(n)$

$B = 4$, $C = 5$, $D = 6$.

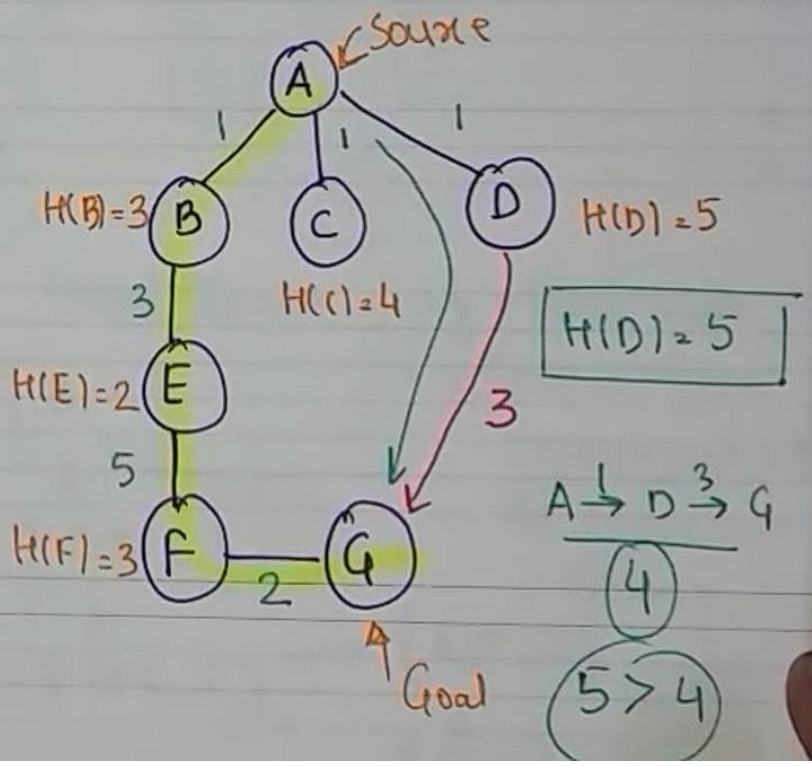
$\leq$

Actual = (11)

$3 < 11$

(ii) **Non-Admissible** :- Overestimate

$H(n) > H'(n)$



Ques:- Differentiate b/w blind and heuristic Search.

Blind Search:- It is also known as unknown/uninformed Search.
- ↳ There is no info^n about the searching.
- ↳ No Knowledge of where the GOAL.
- ↳ Eg:- Depth first, Breadth first Search
- ↳ Efficiency is low
- ↳ Slower than Heuristic
- ↳ Large memory is used
- ↳ No func^n (special) is used.

Heuristic Search:- It is a method of solving problems more easily and fast. They have Knowledge of where goal or finish of the graph. (Informed Search)

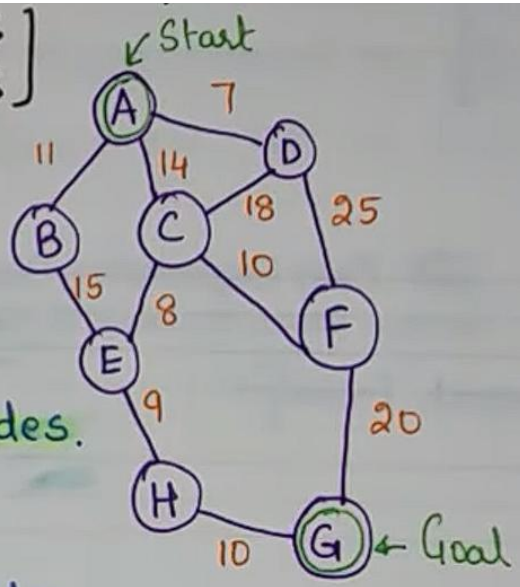
Eg:- Hill climbing, A*, AO*.
- ↳ Highly efficient < less time, Less cost
- ↳ finds sol^n Quickly.
- ↳ no large memory is required.
- ↳ Heuristic func^n is used.

BEST FIRST SEARCH: GREEDY SEARCH. [BFS / DFS]

↳ Uses evaluation Algorithm (func^{n}) to decide which adjacent node is most promising and then explore.

↳ Category of Heuristic or Informed Search.

↳ Priority Queue is used to store Cost of nodes.



ALGORITHM:

→ Sorted order.

Priority Queue 'PQ' Containing initial States

Loop

if PQ = Empty Return FAIL

Else

NODE ← Remove_First(PQ)

if NODE = GOAL

Return Path from Initial to NODE

Else

Generate all Successors of NODE and insert newly generated NODE into PQ according to COST Value.

END LOOP

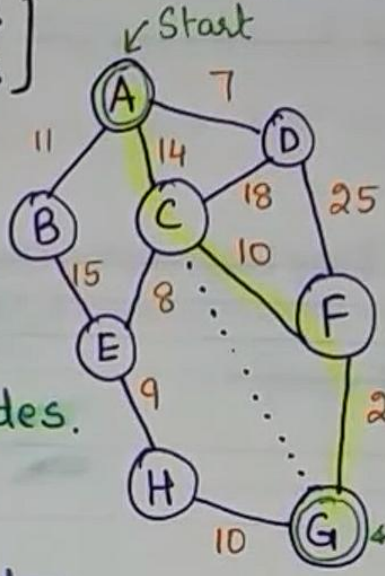
BEST FIRST SEARCH: GREEDY SEARCH. [BFS / DFS]

↳ Uses evaluation Algorithm (func^n) to decide which adjacent node is most promising and then explore.

↳ Category of Heuristic or Informed Search.

↳ Priority Queue is used to store Cost of nodes.

→ Sorted order.



Straight Line distance.

A → G = 40
B → G = 32
C → G = 25
D → G = 35
E → G = 19
F → G = 17
G → G = 0
H → G = 10

ALGORITHM:

Priority Queue 'PQ' Containing initial States

Loop

if PQ = Empty Return FAIL

Else

NODE ← Remove_First(PQ)

if NODE = GOAL

Return Path from Initial to NODE

Else

Generate all Successors of NODE and insert newly generated NODE into PQ according to COST Value.

END LOOP

Open	Closed
[A]	[]
[C, B, D]	[A]
[B, D]	[A, C]
[F, E, B, D]	[A, C]
[E, B, D]	[A, C, F, G]

Path = A → C → F → G

(44)

Space Com:- $O(b^d)$

T.C:- $O(b^d)$

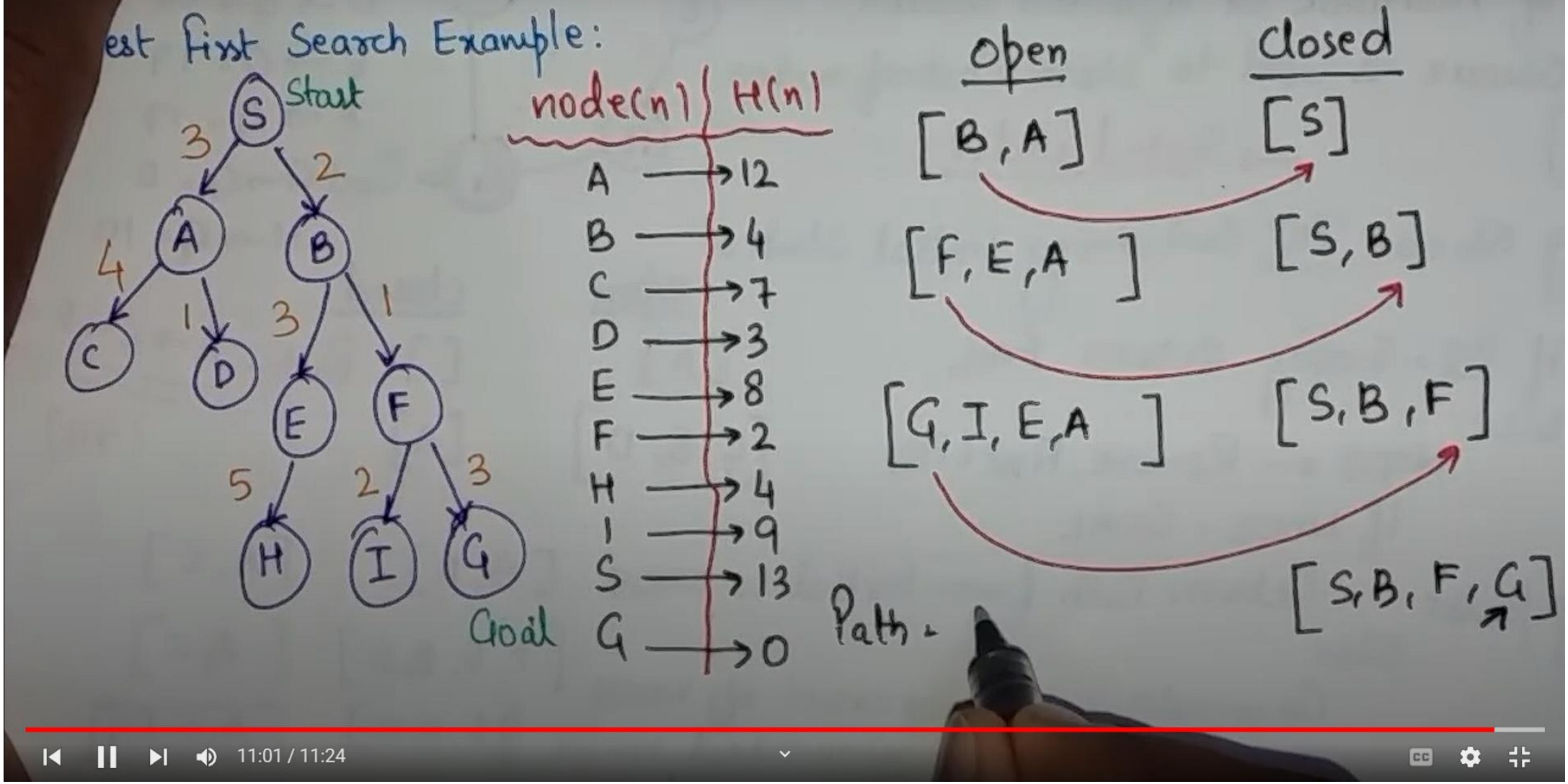
b: branching factor

d: depth

L26: Best First Search(BFS) | Informed Search in Artificial Intelligence with Examples | AI Lectures

Easy Engineering Classes - Free YouTube Lectures

EEC Classes GGSIPU, UPTU, Mumbai Univ., Pune Univ., GTU, Anna Univ., PTU and Others EEC Classes



11:01 / 11:24

Ques:- Explain A* Algorithm for Search.

↳ Uses heuristic func^n $h(n)$ and cost to reach the node 'n' from start state $g(n)$.

↳ Finds shortest path through search space.

↳ Fast and optimal result.

$f(n) = g(n) + h(n)$ ⟶ heuristic Value (Child node)

$f(n)$: estimated cost.

$g(n)$ ↳ Cost to reach node

ALGORITHM:

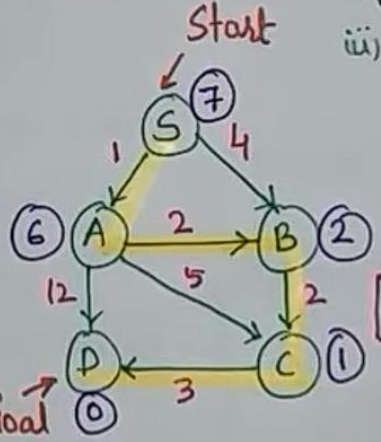
↳ i) Enter starting node in OPEN list.
ii) If OPEN List is Empty return FAIL
iii) Select node from OPEN list which has smallest value (g+h).
↳ if node = Goal, return Success
iv) Expand node 'n' and generate all Successors
↳ Compute (g+h) for each Successor node.
v) if node 'n' is already in OPEN/CLOSED, attach to back pointer.
vi) Go to (ii)

Advantages:-

↳ i) Best searching algorithms.
ii) optimal and complete.
iii) Solving Complex problems.

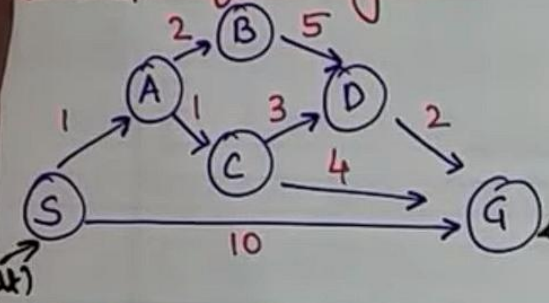
Disadvantages:-

↳ i) Doesn't always produces shortest.
ii) Complexity issues.
iii) Requires memory.



$S \to A = 1+6 = 7$ ✓
$S \to B = 4+2 = 6$ ✓
$S \to B \to C = 4+2+1 = 7$ ✓
$S \to B \to C \to D = 4+2+3+0 = 9$ r1
$S \to A \to B = 1+2+2 = 5$ ✓
$S \to A \to C = 1+5+1 = 7$ ✓
$S \to A \to D = 1+12 = 13$ r2
$S \to A \to B \to C = 1+2+2+1 = 6$ ✓
(#) $S \to A \to B \to C \to D = 1+2+2+3 = 8$
$S \to A \to C \to D = 1+5+3 = 9$ r4

Example of A* Algorithm:-



State	h(n)
S	5
A	3
B	4
C	2
D	6
G	0

$S \to A = 1+3 = 4$

$S \to G = 10+0 = 10$

$S \to A \to B = 1+2+4 = 7$ ✓

$S \to A \to C = 1+1+2 = 4$ ✓

$S \to A \to C \to D = 1+1+3+6 = 11$ ✓

$= S \to A \to C \to G = 1+1+4 = 6$.

$S \to A \to B \to D = 1+2+5+6 = 14$ =

$S \to A \to C \to D \to G = 1+1+3+2 = 7$

$S \to A \to B \to D \to G = 1+2+5+2 = 10$

Ques:- Describe how Problem Reduction is done with AO* Algorithm.

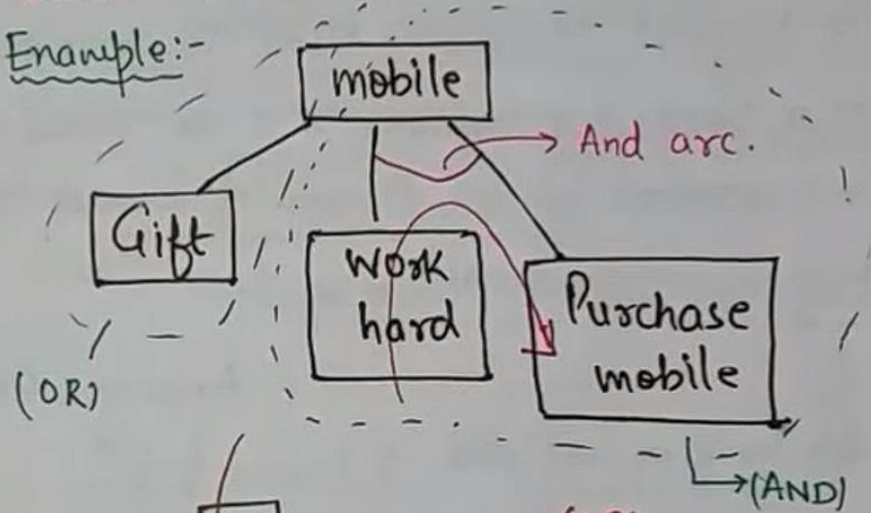
AND-OR Graphs:- useful for representing the problem solution that can be solved by decomposing them into smaller set of problems, all of which must be solved.

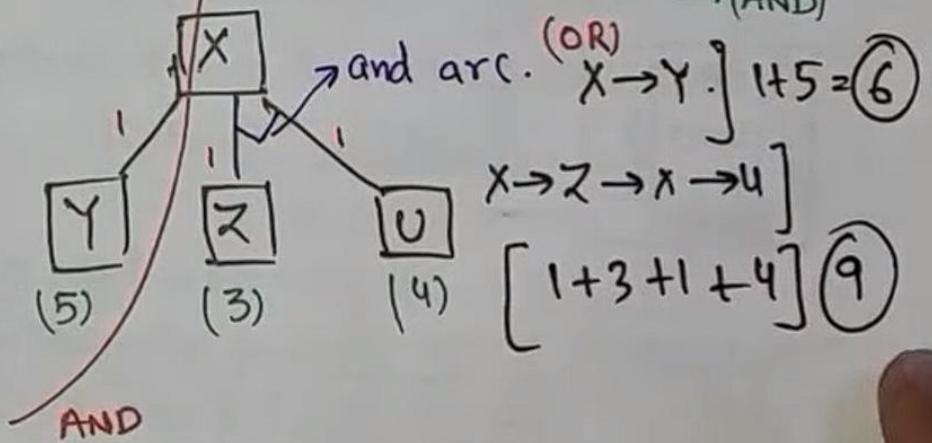
→ ANDARCS. } → May point to 'n' no. of successor nodes.

↳ Also, like Best-Fit Search can be used that has the ability to handle AND Arcs.

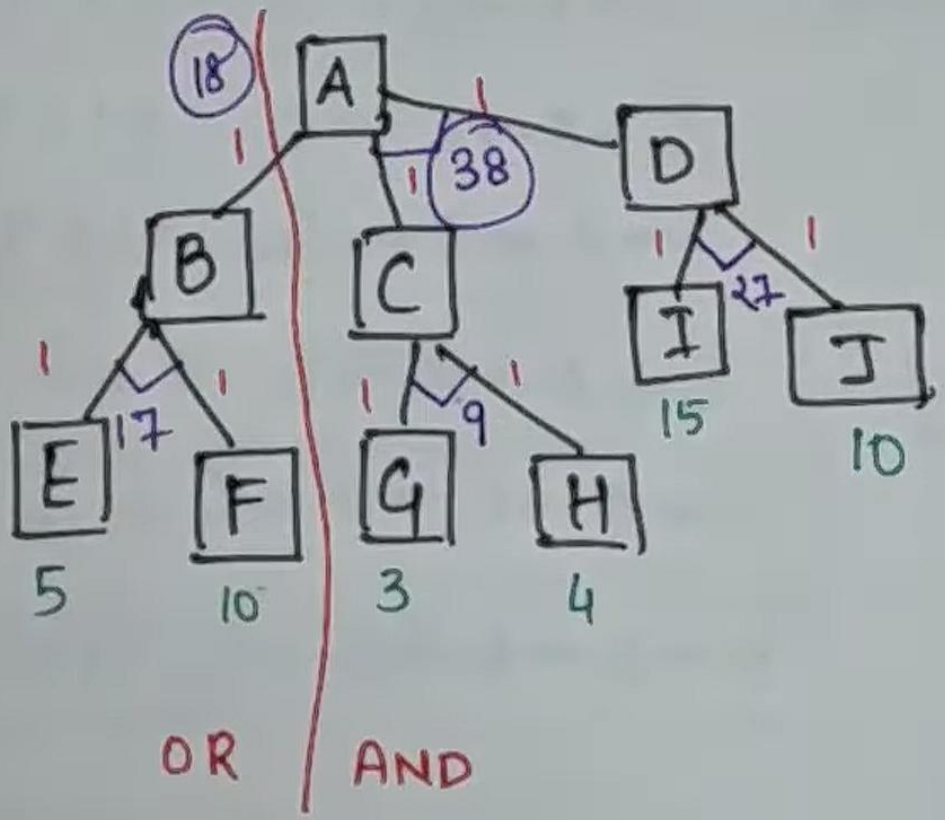
[FUTILITY: Should be chosen to correspond to a threshold value. if Est. Cost > Futility. Stop Search.]

Example:-





AO* Algo:-



L30: Hill Climbing Search in Artificial Intelligence | Limitation of Hill Climbing Search in AI

Ques:- Write Short note on "Hill-Climbing Search" → Local search algo, → Greedy Approach] No Backtracking.

↳ Variant of generate and test method in which feedback from test procedure is used to help generator decide which dir^n to move in Search Space.] always moves in a Single dir^n.

↳ It is like DFS }

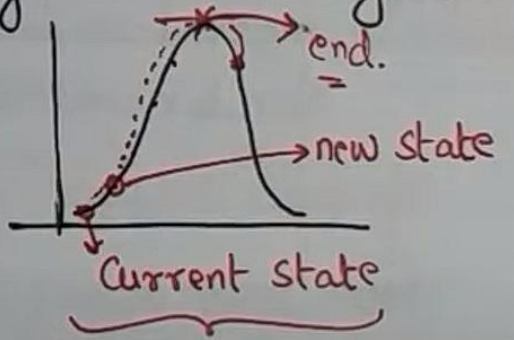
Eg:-

1	2	4
5		7
3	6	8

(Starting State)

(4) (5) (X) (6)

↳ we will choose this

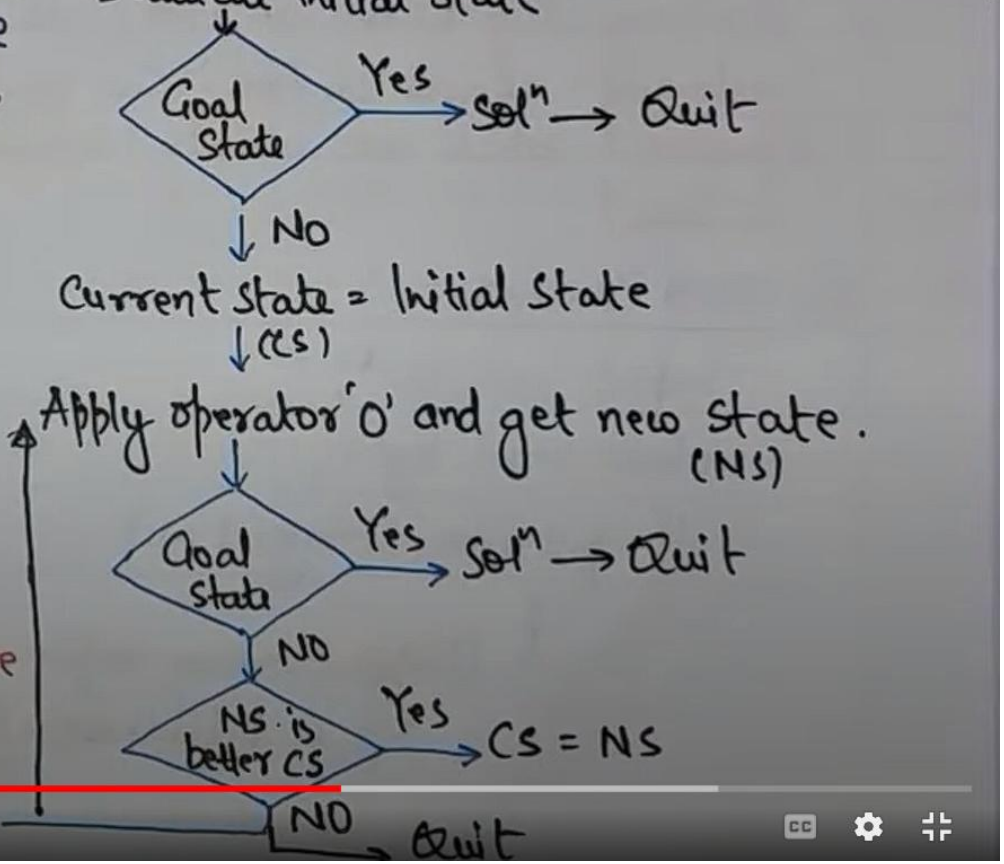


if new state is better than Current state.

⇓

new state = Current state

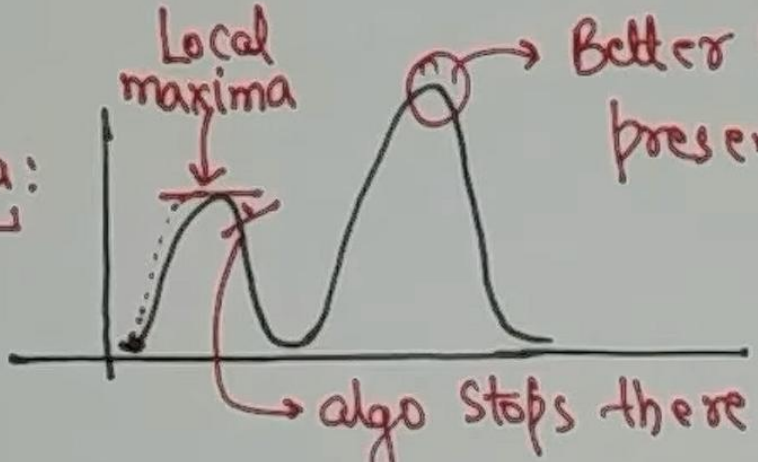
Flowchart :-



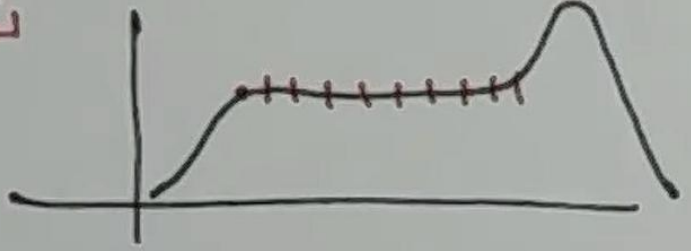
8:21 / 11:55

Limitations:-

(1) Local maxima:

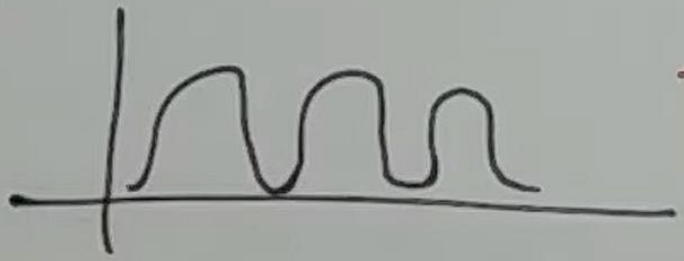


(2) Plateau:



algo gets Same Value/ Soln for some time.

(3) Ridge:-



all Successor gets Same Value.

L34: State Space Search | Eight Tile Puzzle Problem | Artificial Intelligence Lectures in Hindi

Easy Engineering Classes - Free YouTube Lectures

EEC Classes GGSIPU, UPTU, Mumbai Univ., Pune Univ., GTU, Anna Univ., PTU and Others EEC Classes

State Space Search: Used in Problem Solving.

↳ It is a process used in A.I in which Successive Configurations or States of an instance are Considered with intention of finding a GOAL State with desired property.

↳ Problems are modelled as State Space

↳ Representation:

S: (S, A, Action(S), Result(s,a), Cost(s,a))

- S: set of all possible states.
- A: Set of all possible actions
- Action(S): Func^n [which action is possible for Current State]
- Result(s,a): Func^n [State reached by performing action 'a' on state 's']
- Cost(s,a): Costing.

set of states in which a problem Can be.

EIGHT TILE PUZZLE

Start(S)

1	4	3
2		5
8	6	7

GOAL(G)

1	2	3
8		4
7	6	5

Actions Possible:- up, down, Left, Right

Legal moves for a state.

'Right'

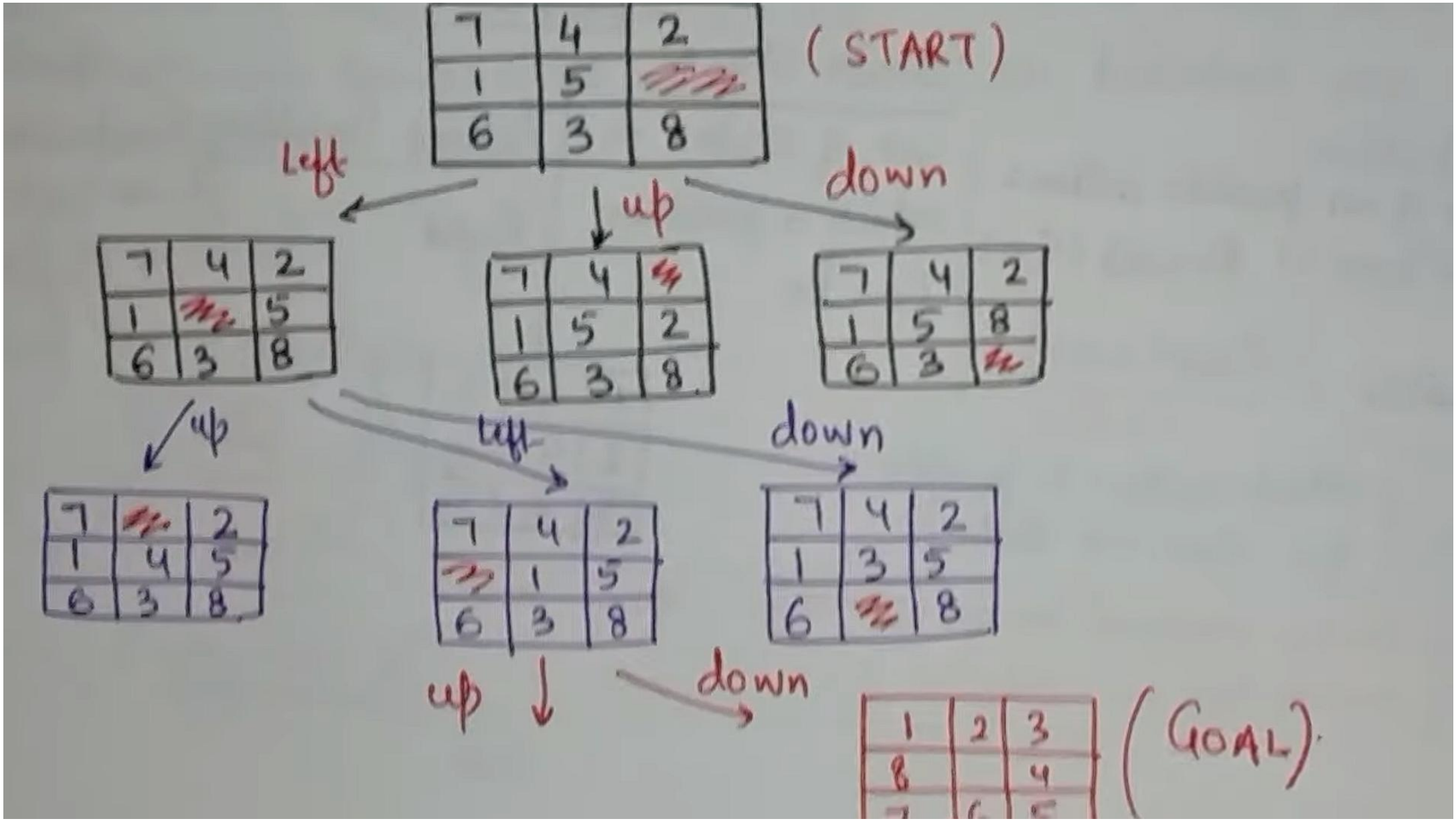
1	4	3
2	5	
8	6	7

new State(S1)

right is not possible now

up, down, left

9:35 / 13:48


7 4 2
1 5
6 3 8
(START)
left
up
down
7 4 2
1 5
6 3 8
7 4
1 5 2
6 3 8
7 4 2
1 5 8
6 3
up
left
down
7 2
1 4 5
6 3 8
7 4 2
1 5
6 3 8
7 4 2
1 3 5
6 8
up
down
1 2 3
8 4
(GOAL)

Missionaries and Cannibals Problem:

3 Missionaries and 3 Cannibals were on
Side of the river.
→ All want to cross the river.
→ On Same Side of river missionaries
ount can't be less than Cannibals.
→ Only one boat available that can hold
Only two people at a time.

State Representation: <R, M, C>

- R: River side (1 (Starting), 2 (Ending))
- M: no. of missionaries
- C: no. of Cannibals.

<1,3,3> <2,0,0>] Initial State
↓
<1,3,1> <2,0,2>
↓
<1,3,2>, <2,0,1>
↓
<1,3,0> <2,0,3>
↓
<1,3,1>, <2,0,2>
↓
<1,1,1> <2,2,2>
<1,2,2>, <2,1,1>
↓
<1,0,2> <2,3,1> → <1,0,3>, <2,3,0>
↓
<1,0,1> <2,3,2>
← <1,0,2> <2,3,1>
← <1,0,0>, <2,3,3>

TIC-TAC-TOE PROBLEM: {Zero-Kata}

Two-Player Game where one player called MAX marks a letter 'X' and the opponent called MIN marks a letter 'O'.

↳ '3×3' Grid where two players put their letters.

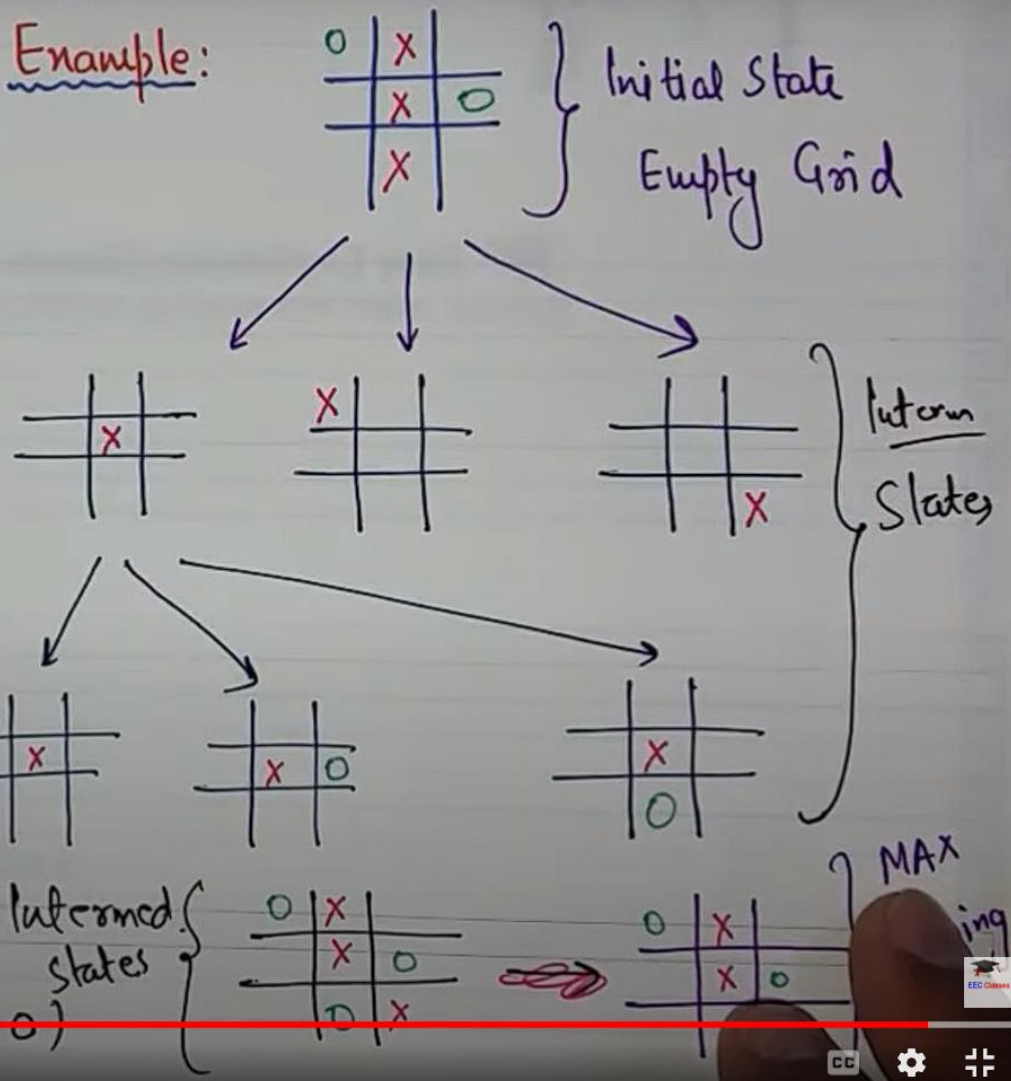
↳ Player with Same marks (three) in Complete ROW/COLUMN/DIAGONAL wins the Game.

↳ Start State = Empty Grid.

↳ Goal State = Win for either of player

MAX (3 Consecutive X) MIN (3 Consecutive O)

Example:



Travelling Salesman Problem and its State Representation.

Salesman has list of cities, each of which must be visited exactly once. Starting point.

↳ In List, there are direct roads b/w each pair of cities.

↳ To find Route, Salesman should follow shortest possible round trip.

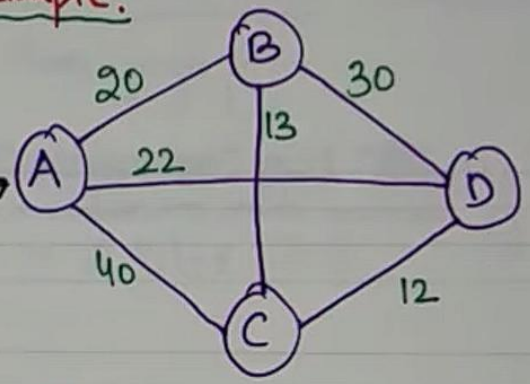
↳ State is represented as pair of any two cities and distance b/w them.

State Space:- Initial State (State A)

↳ {(A $\xrightarrow{20}$ B), (A $\xrightarrow{40}$ C), (A $\xrightarrow{22}$ D)}

State

Example:



Possible Routes:-

(A $\xrightarrow{20}$ B $\xrightarrow{13}$ C $\xrightarrow{12}$ D $\xrightarrow{22}$ A) = 67 ✓

(A → B → D → C → A) = 102

(A → D → B → C → A) = 105

(A → D → C → B → A) = 67 ✓

(A → C → D → B → A) = 102

(A → C → B → D → A) = 105