

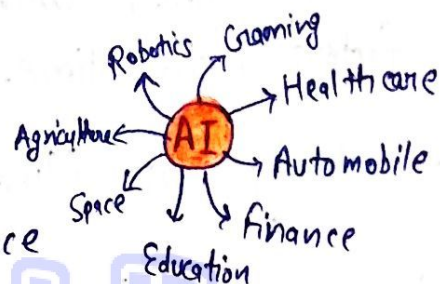
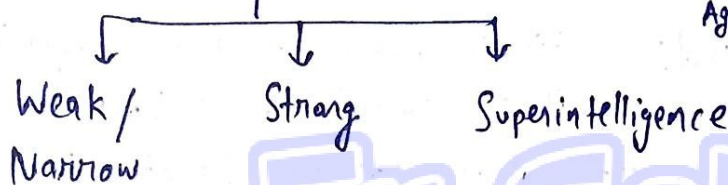
Artificial Intelligence

manmade thing

Thinking power

→ AI refers to simulation of human intelligence in machines that are programmed to think like humans & mimic their actions.

Type of AI



Components of AI ⇒

- Applications :— Image Recognition
Speech Recognition
Chatbots
Natural language generation
Sentiment Analysis
- Types of Models :—
Deep learning
Machine learning
Neural Networks
- SW & HW for training & running models :—
GPUs, parallel processing tools (like spark)
cloud data storage & compute platforms
- Programming languages :— Python, TensorFlow, Java, C

Different approaches of AI

Reactive Machine

Limited Memory

Theory of Mind

Self-Awareness

1. Reactive Machine ⇒ These machines are most basic form of AI application
Eg - Deep Blue, IBM's chess-playing supercomputer.

AI teams do not use any training sets to feed machines nor do latter store data for future references. Based on move made by opponent, the machine decides next move.

2. **Limited Memory**:- These machines belong to class II category of AI applications. Self-driven cars are perfect example. These machines are fed with data & are trained with other cars speed & direction, lane markings, traffic lights, curves of roads, and other important factors, over time.

3. **Theory of Mind**:- This is where we are, struggling to make this concept work, however, we are not there yet. TOM is concept where bots will be able to understand human emotions, thoughts & how they react to them, if AI-powered machines are ever to mingle with us & move around with us, understanding human behavior is imperative. And then, reacting to such behavior accordingly is the requirement.

4. **Self-Awareness**:- It is extension of class III type of AI. It is one step ahead of understanding human emotions. This is phase where building self aware machines seem far-fetched from where we stand today.
eg- Where someone is honking from behind, the machines should be able to feel the emotion. That's when they understand how it feels when they honk at someone back.

Problem solving \Rightarrow Chess, TSP, TOH, N-Queen, AI Agent \rightarrow $\begin{cases} \text{searching} \rightarrow \text{info.} \\ \text{env.} \end{cases}$

Define the problem \Rightarrow Analyzing the problem \Rightarrow Identification of problem
 \Downarrow
Change the solution
 \Leftarrow Implementation

Searching

Uninformed / Blind

- Breadth first Search
- Uniform cost search
- Depth first search
- Depth limited Search
- Iterative Deeping depth first Search
- Bidirectional Search

Informed / Heuristic

- Best First Search
- A* search
- AO* Algorithm
- Problem Reduction
- Hill Climbing

Uninformed / Blind Search:- It doesn't contain any domain knowledge such as closeness, location of goal. It operates in Brute force way, as it only includes info. about how to traverse the tree & how to identify leaf & goal nodes.

Informed Search:- It uses domain knowledge, the problem inf. is available which can guide the search.

A heuristic is a way which might not always be guaranteed for best soln but guaranteed to find a good soln in reasonable time.

Breadth First Search (BFS):- Traversing a tree

FIFO Queue

Time Complexity:- $b^0 + b^1 + b^2 + \dots + b^d$

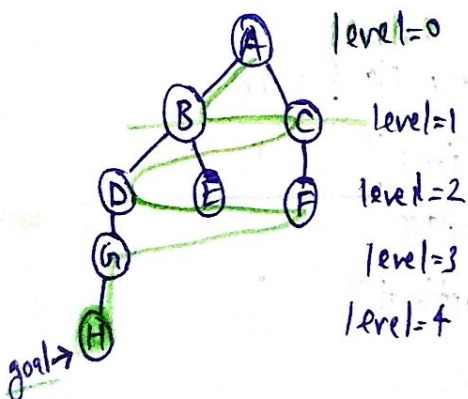
$b < \text{max. branching}$

$d = \text{depth}$

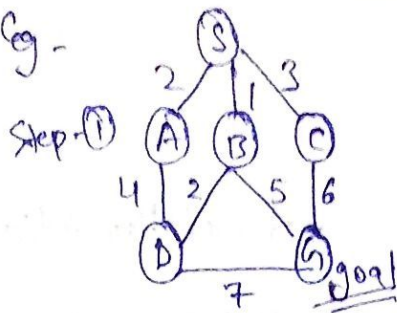
$1 + b + \dots + b^d$

$O(b^{d+1})$

Space:- $O(bd)$

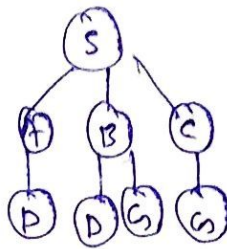


eg -



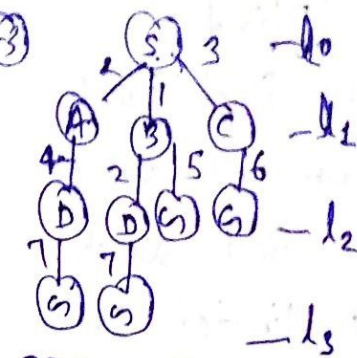
Step-1

Step-2



find route using BFS?

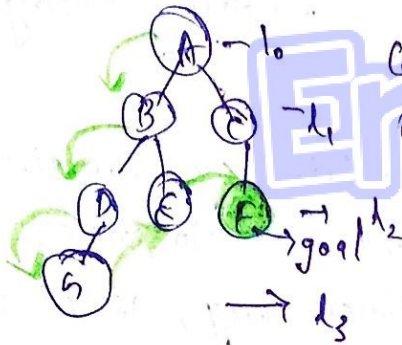
Step-3



SBC₆ on SCG₇

Depth First Search (DFS):

Stack LIFO



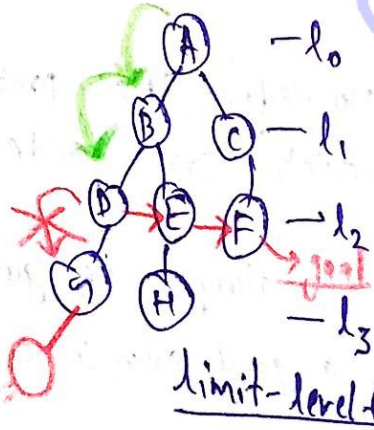
Complete? No (infinite loop)

Optimal? No

Time complexity? $O(b^d)$

Space? $O(b \times d) = O(bd)$

Depth Limited Search (DLS):

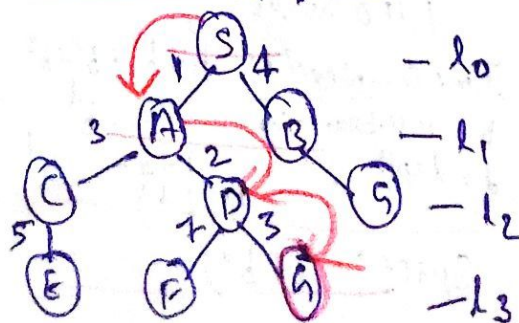


Complete? No

Optimal? No (more than one soln)

{ memory efficient }
adv.

Uniform Cost Search:



Complete?

Time?

Space?

Optimal?

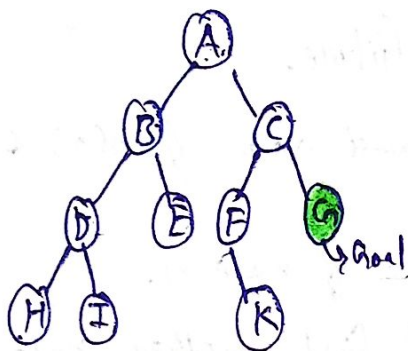
Iterative Deepening DFS: - It is a combination of DFS & BFS.

→ This algo. performs depth-first search up to a certain "depth-limit" & it keeps increasing the depth limit after each iteration until the goal node is found.

Adv. → This algo. combines the benefit of

BFS's fast search & DFS's memory efficiency.

Dis. → It repeats all work of previous phase.



level 0 DL=0 I₁: A

DL=1, I₂: A, B, C

level 1

DL=2, I₃: A, B, D, E, C, E, G

level 2

DL=3 I₄: - A, B, D, H, I, E, C, F, K, G ✓

level 3

4th ✓

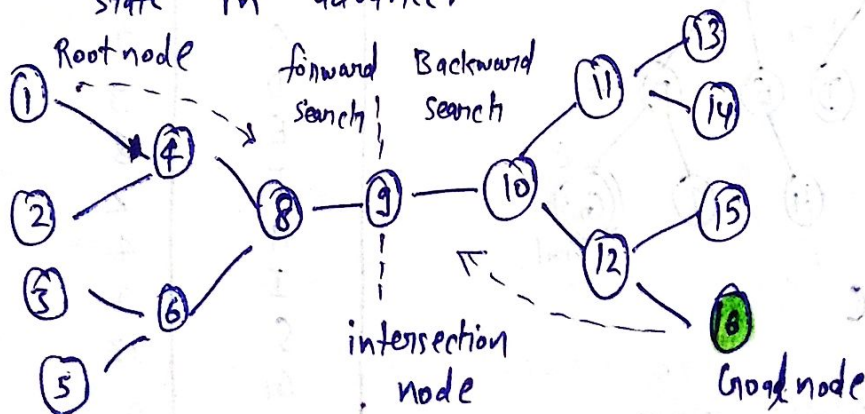
Bi Directional Search: - It does forward search & backward search.

It uses BFS, DFS, DLS etc. → One start from start node and other ^{from} goal node.

Adv. → fast, less memory

Dis. → Implementation is difficult. → Stop when intersect.

→ One should know the goal state in advance.



Best First Search [Greedy] :- It is combination of DFS & BFS.

→ It uses heuristic function $h(n) \leq h^*(n)$ and

$h(n)$ = heuristic cost

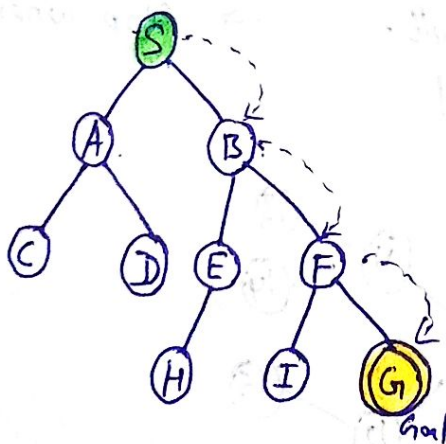
$h^*(n)$ = estimated cost

→ It is implemented by priority queue.

BFS Algo. ⇒

1. Place starting node into open list
2. If open list is empty, stop & return failure.
3. Remove node n from open list which has lowest value of $h(n)$ & place it in closed list.
4. Expand node n & generate successors of node n .
5. Check each successor of node n and find whether node is goal node or successor node is goal node return it.
6. For each successor node find $f(n)$ & node that has not been in both list, add in open list
7. Return to step (2)

Eg.



node	H(n)
A	12
B	4
C	7
D	3
E	8
F	2
H	4
I	9
S	13
G	0

Initialization :- Open[A, B] close[S]

Iteration 1 :- Open[A] close[S, B]

Iteration 2 :- Open[E, F, A] close[S, B] ⇒ Open[E, A] close[S, B, F]

Iteration 3 :- Open[I, G, E, A] close[S, B, F] ⇒ Open[I, E, A] close[S, B, F, G]

A* search Algorithm :- It is most commonly known form of best-first search.

→ It uses $h(n)$ & cost to reach the node n from start state $g(n)$.

→ It has combined features of UCS & greedy best-first Search.

$$\text{Estimated Total cost} \leftarrow f(n) = g(n) + h(n)$$

\downarrow Cost to reach node n from start state \downarrow Cost of path from n to goal

Step - (1) Place starting node in Open list.

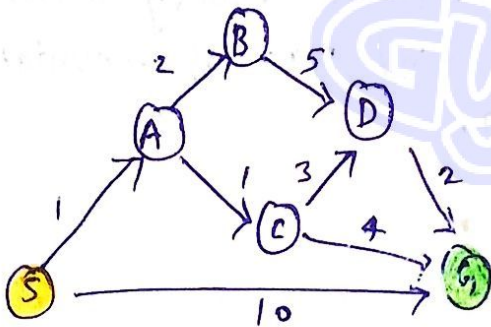
Step - (2) Check if open list is empty.

Step - (3) Select node from open list which has smallest value of evaluation function ($g+h$)

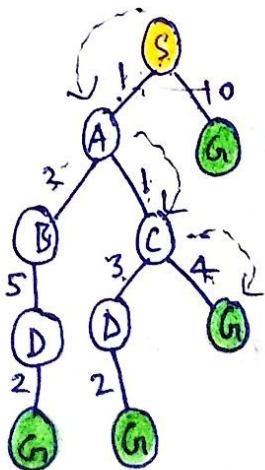
Step - (4) Expand n and generate all of its successors & put n into closed list.

Step - (5) If node n_i is already in open & closed list, then it should be attached to back pointer which reflects lowest $g(n)$ value.

Step - (6) Return to step - (2)



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



Initial $\{S, 5\}$ $0+5=5$

Iteration-1:- $\{(S \rightarrow A, 4), (S \rightarrow G, 10)\}$

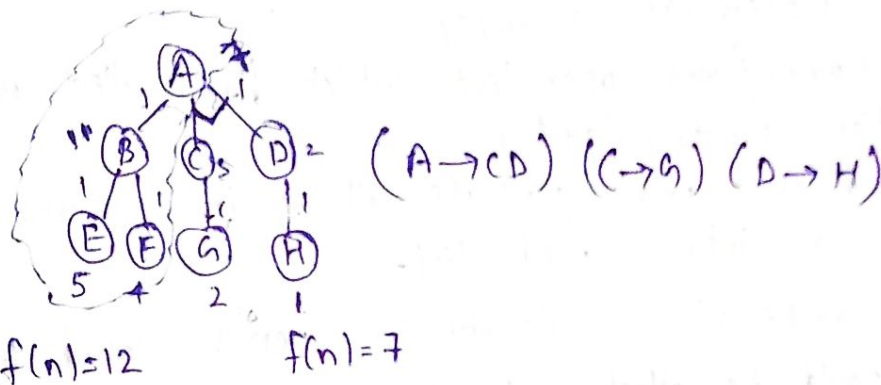
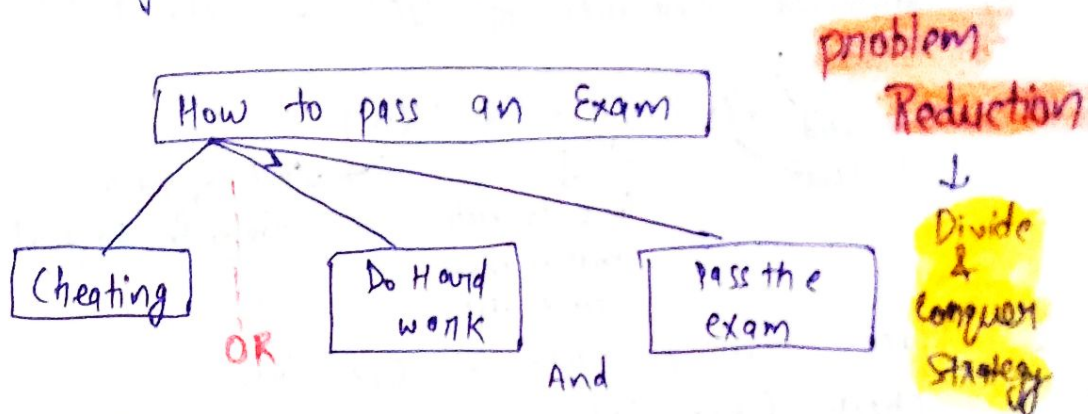
I_2 :- $\{(S \rightarrow A \rightarrow B, 7), (S \rightarrow A \rightarrow C, 4), (S \rightarrow G, 10)\}$

I_3 :- $\{(S \rightarrow A \rightarrow C \rightarrow D, 11), (S \rightarrow A \rightarrow C \rightarrow G, 6), (S \rightarrow G, 10), (S \rightarrow A \rightarrow B, 7)\}$

$S \rightarrow A \rightarrow C \rightarrow G, (6)$

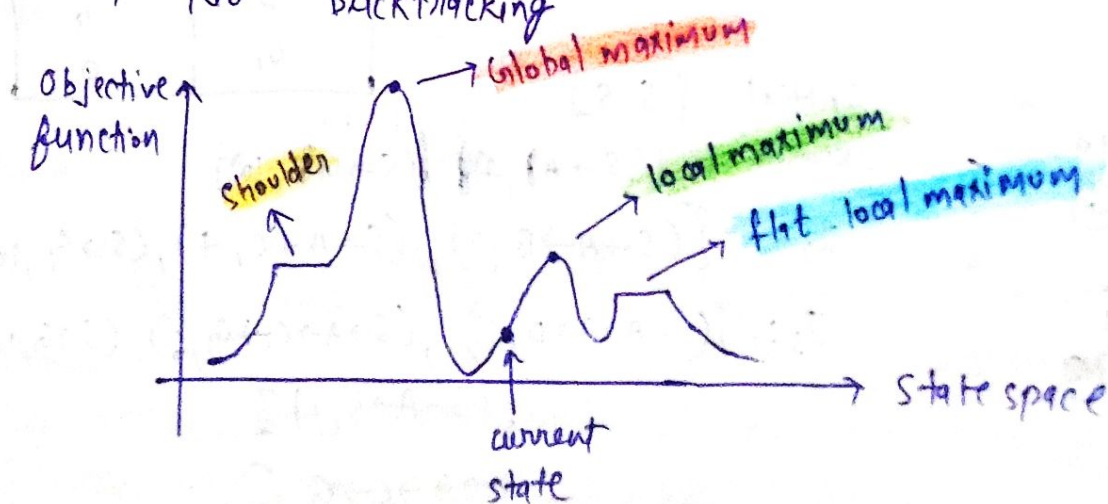
AO* Search Algorithm:- It is used to solve AND-OR graphs.

It is basically based on problem decomposition.

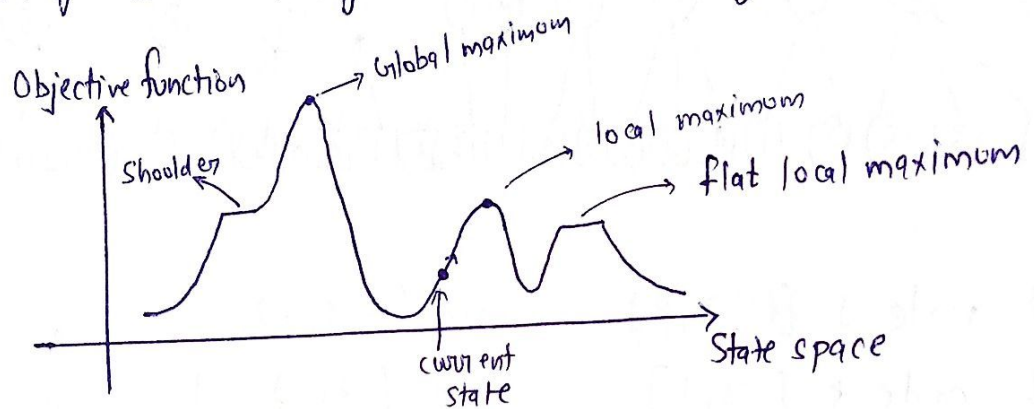


Hill Climbing Algorithm \Rightarrow It is a local search algo. which continuously moves in direction of increasing value to find peak of mountain or best solution to the problem.

- \rightarrow Generate & Test variant
- \rightarrow Greedy approach
- \rightarrow No backtracking



→ Hill Climbing is mostly used when a good heuristic is available.



Types of Hill Climbing:-

- (i) Simple Hill Climbing:- It only evaluates the neighbor node state at a time & selects the first one which optimizes current cost & set it as a current state.
- (ii) Steepest-Ascent Hill Climbing:- This algorithm examines all the neighboring nodes of the current state & selects one neighbor node which is closest to the goal state. This algorithm consumes more time as it searches for multiple neighbors.
- (iii) Stochastic hill climbing:- It does not examine for all its neighbors before moving. Rather, this search algo selects one neighbor node at random & decides whether to choose it as a current state or examine another state.

Constraint Satisfaction Problem [CSP] \Rightarrow

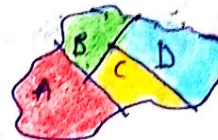
- \rightarrow CSP consists of 3 components V, D, C
- $\rightarrow V$ is set of variables $\{v_1, v_2, \dots, v_n\}$
- $\rightarrow D$ is set of Domains $\{D_1, D_2, \dots, D_n\}$ one for each variable.
- $\rightarrow C$ is set of constraints that specify allowable combination of values
 $C_i = (\text{scope}, \text{rel})$



cryptarithmic

Two
+ Two
FOUR

$$D = \{R, B, G\}$$
$$V = \{A, B, C, D\}$$



$$C_1 = \{(A, B), A \neq B\}$$

$$C_2 = \{(A, B), (R, G), (G, R), (R, B), (G, B), (B, R)\}$$

$A_1 - (B)(i)$

Mini-Max Algorithm :-

(5)

- \rightarrow It is a recursive on backtracking algorithm which is used in game theory and decision-making.
- \rightarrow It is mostly used for game playing in AI - Such as Chess, tic-tac-toe, go, checkers etc.
- \rightarrow There are 2 players **MAX & MIN**.
- \rightarrow Max for maximized value & MIN for minimized value.
- \rightarrow minimax performs a DFS algorithm

Properties of Mini-Max :-

Complete - Yes

Optimal - Yes

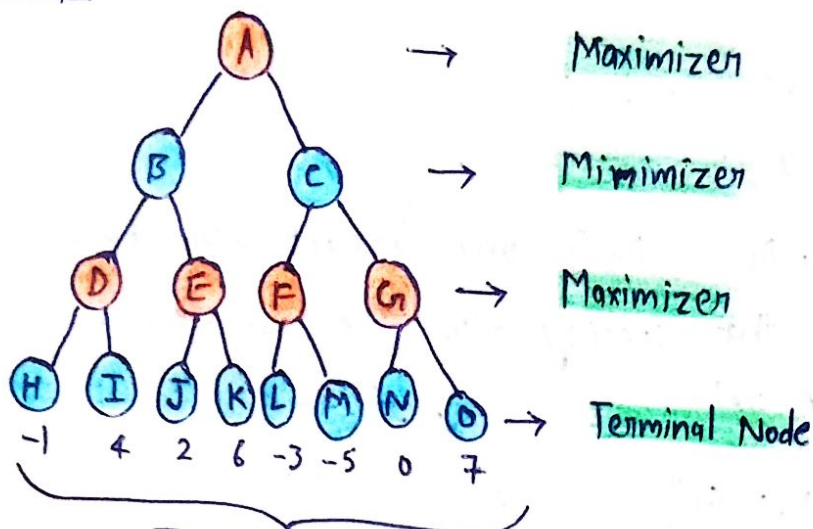
Time Complexity - $O(b^m)$

Space Complexity - $O(bm)$

Limitation :-

It is slow for complex games such as chess, go etc. coz these have branching factor. So this limitation of minimax can be improved from **alpha-beta pruning**.

Step - 1



Maximizer = $-\infty$

Minimizer = $+\infty$

MiniMax Algorithm

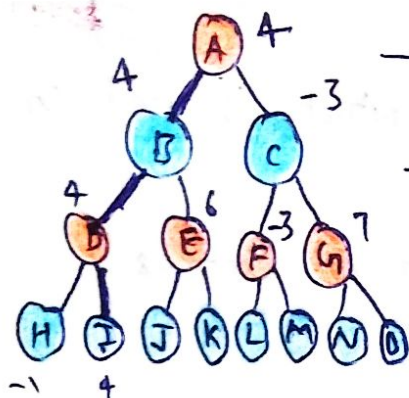
(6)

At node D $\{-1, 4\}$ $\text{Max}(-1, -\infty) = 4 = \text{max}(-1, 4)$

At node E $\{2, 6\}$ $\text{Max}(2, -\infty) = 6 = \text{max}(2, 6)$

At F $\{-3, -5\}$ $\text{Max}(-3, -\infty) = -3 = \text{max}(-3, -5)$

At G $\{0, 7\}$ $\text{Max}(0, -\infty) = 7 = \text{max}(0, 7)$



Node B $\Rightarrow \min(4, 6) = 4$

Node C $\Rightarrow \min(-3, 7) = -3$

maximizer
Node A :-

$\text{max}(4, -3) = 4$

(ii)

Alpha-Beta Pruning \Rightarrow

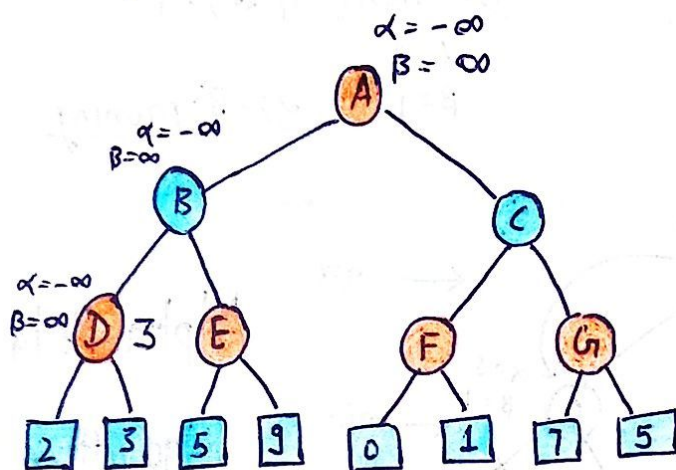
→ It is a modified version of minimax algorithm. It is an optimization technique for minimax algorithm.

→ There is a technique by which without checking each node of game tree we can compute correct minimax decision and this tech. is called pruning. This involves 2 threshold parameter Alpha & beta for future expansion. so it is called Alpha-Beta pruning.

Alpha :- The best (highest value) choice we have found so far at any point along the path of maximizer. (Max player)
The initial value of alpha is $-\infty$.

Beta :- The best (lowest value) ——— of minimizer. $+\infty$. (Min player)

→ It removes all the nodes which are not really affecting the final decision.



→ max

→ min

→ max

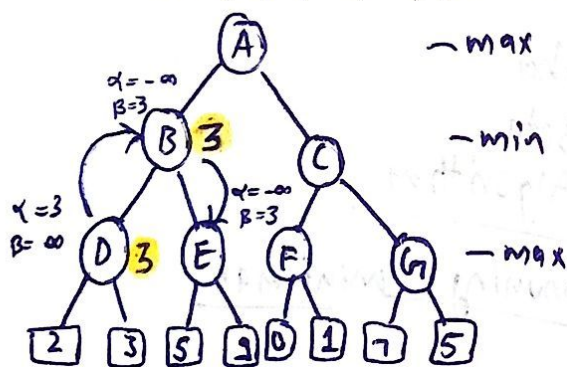
→ Terminal node

Alpha Beta
Algorithm

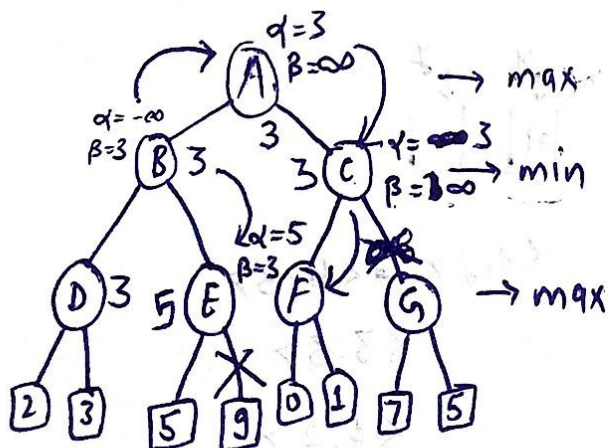
→ while Backtracking, node values will be passed to upper nodes instead of values of alpha & beta.

→ We will only pass alpha, beta values to child nodes.

$\max(2, 3) = 3$, so node D value will also 3, $\alpha = 3$



$\alpha = -\infty, \beta = \min(3, \infty) = 3$



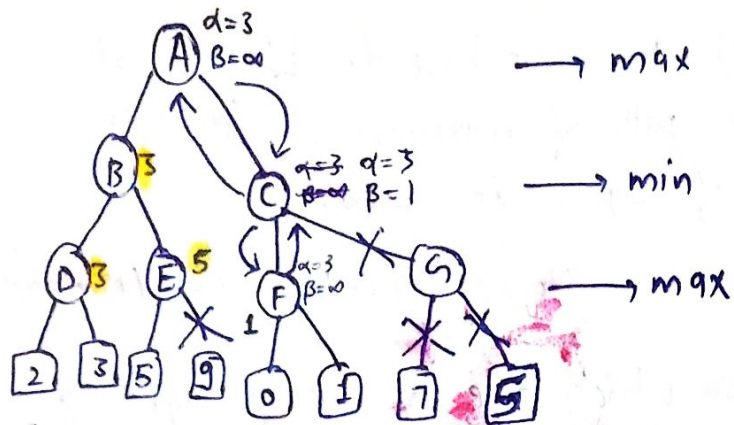
$\max(-\infty, 5) = 5 = \text{node}$

$\alpha = 5, \alpha > \beta$

B to A, backtracking

$\alpha = \max(-\infty, 3) = 3$

$\alpha = 3$



C to F
 $\max(0, 3) = 3$
 $\max(1, 3) = 3$

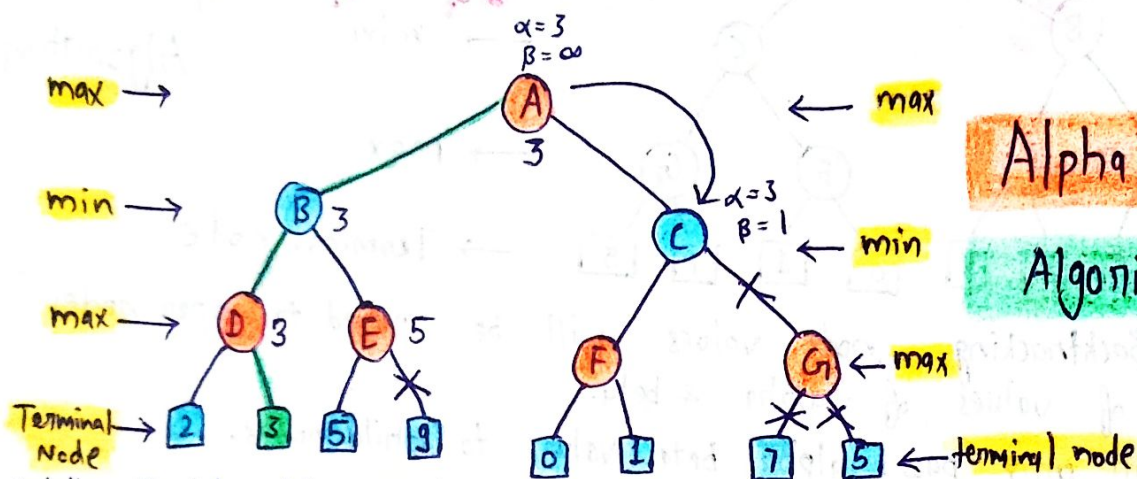
F to C

$\beta = \min(1, \infty) = 1$

$\alpha=3$
 $\beta=1$

$\alpha > \beta$ pruning

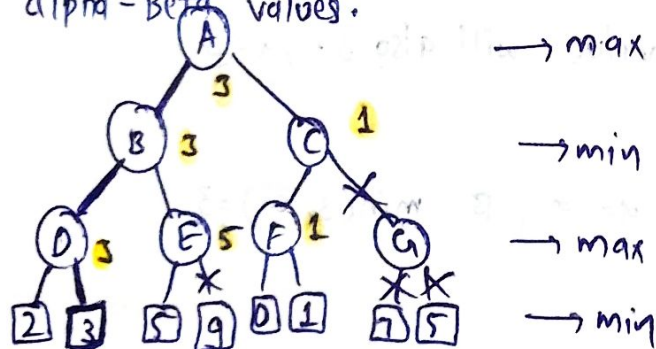
8



Alpha Beta

Algorithm

→ While Backtracking, node values will be passed to upper nodes instead of alpha-Beta values.

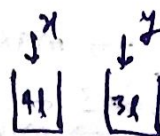


Alpha Beta Algorithm

pruning min-max

WATER JUG PROBLEM :-

Two jug of different capacities are given:



No marking is there on any Jug.

$\langle x, y \rangle = \langle 0, 3 \rangle$

$\langle 3, 3 \rangle$

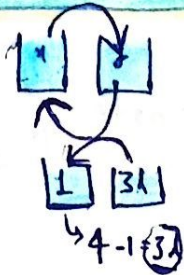
$\langle 4, 2 \rangle$

$\langle 0, 2 \rangle$

$\langle 2, 0 \rangle$

WATER JUG PROBLEM

State space :- $x=4$
 $y=3$



1. $(x, y) \Rightarrow (4, y)$
2. $(x, y) = (x, 3)$
3. $(x, y) = (x-d, y)$, $x > 0$
4. $(x, y) = (x, y-d)$, $y > 0$
5. $(x, y) = (0, y)$
6. $(x, y) = (x, 0)$
7. $(x, y) = (4, y-(4-x))$, $y > 0, x+y > 4$
8. $(x, y) = (x-(3-y), 3)$, $x > 0, x+y > 3$
9. $(x, y) = (x+y, 0)$, $x+y \leq 4$
10. $(x, y) = (0, x+y)$, $x+y \leq 3$ (2, 1)

x	y	Rule
0	0	—
0	3	2
3	0	9
3	3	2
4	2	7
0	2	5
2	0	9

Tile Problem / N-Puzzle Problem \Rightarrow

\rightarrow N puzzle the consists of N tiles (N+1) where N is 8, 15, 24 and soon
If $N=8$ then $8+1=3$ rows and 3 columns

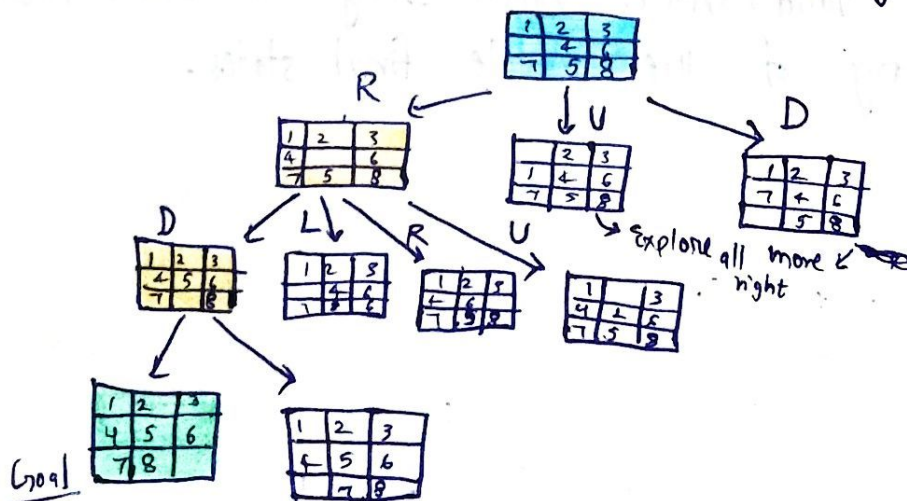
Initial state

1	2	3
	4	6
7	5	8

Goal state

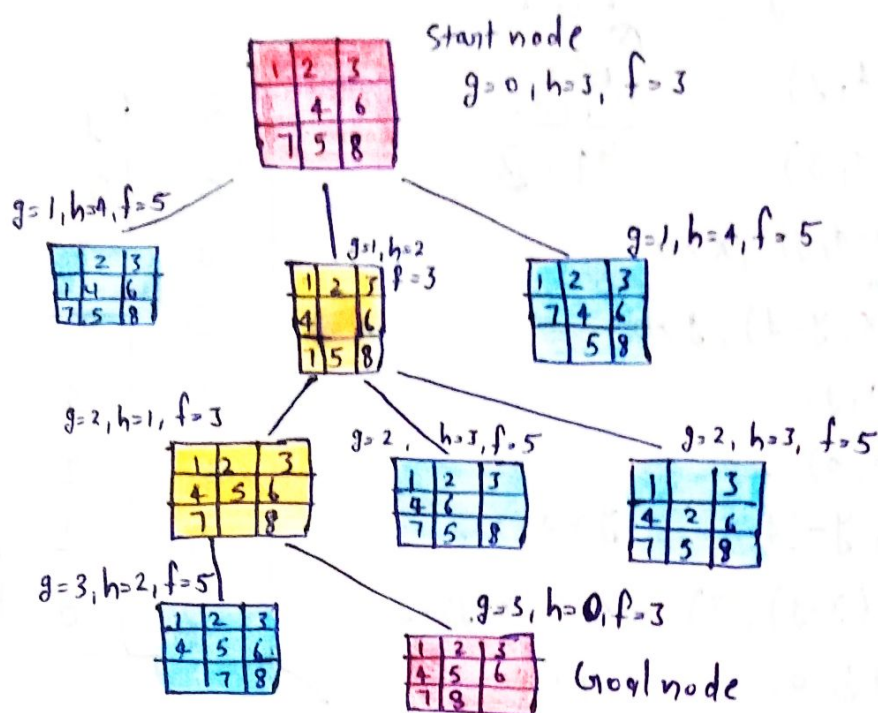
1	2	3
4	5	6
7	8	

\rightarrow Empty space can move in up, down, right & left.



Using
Uninformed
Search

Cost function $(f) = g + h$ Using Heuristic Search



8-puzzle problem

Chess Problem

In a chess game problem, the **start state** is the initial configuration of chessboard. The final or **goal state** is any board configuration, which is a winning position for any player (there may be multiple final positions and each board configuration can be thought of as representation a state of the game). Whenever any player moves any piece, it leads to different state of game. It is estimated that chess game has more than 10^{20} possible states. The game playing would mean finding a sequence of valid moves which bring the board from start state to any of the possible final states.

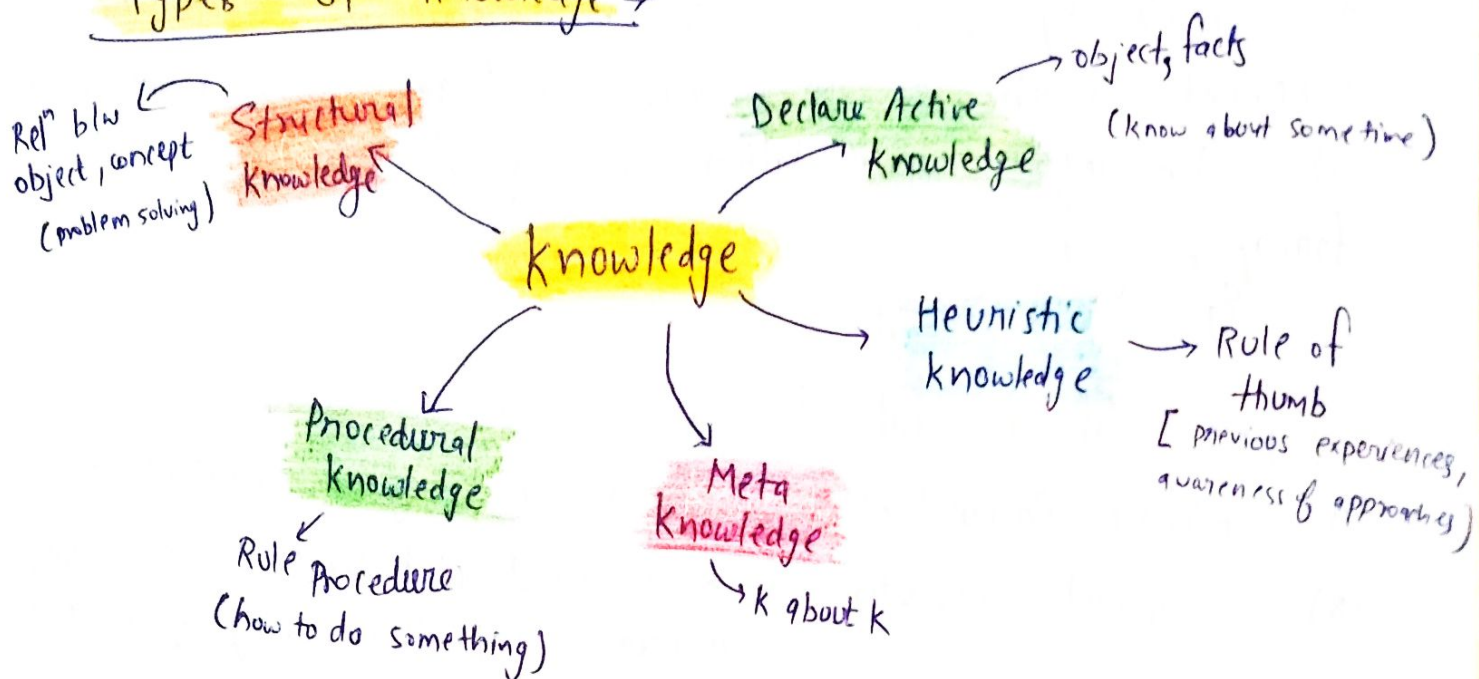
Knowledge Base Representation & Reasoning ⇒

Knowledge Based Agent:-

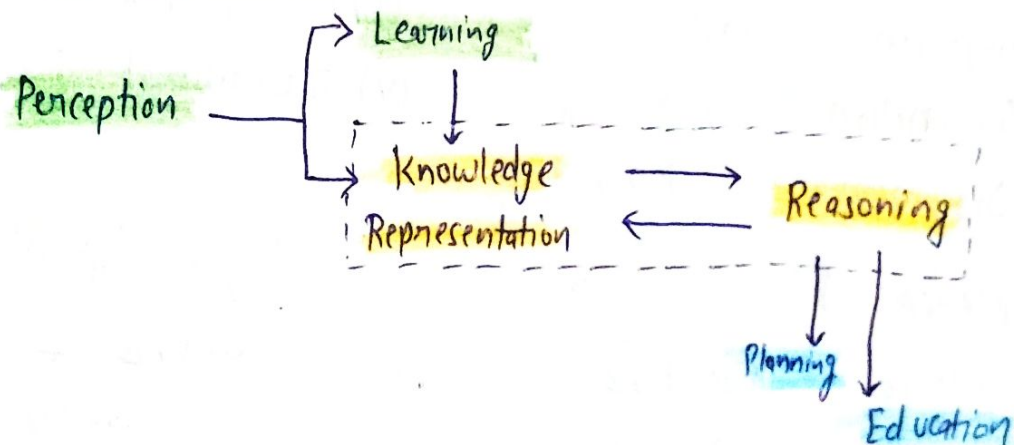
It is those agents who have capability of maintaining an internal state of knowledge, reason over that knowledge, update their knowledge after observations and take actions.

→ knowledge Base is required for updating knowledge for an agent to learn with experiences and take action as per the knowledge.

Types of knowledge ⇒



AI knowledge cycle:-



Approaches of knowledge Representation:-

- (i) Simple Relational knowledge
- (ii) Inheritable knowledge
- (iii) Inferential knowledge
- (iv) Procedural knowledge

Propositional Logic in Artificial Intelligence \Rightarrow

It is the simplest form of logic where all the statements are made by propositions. A proposition is a declarative statement which is either true or false. It is a technique of knowledge representation in logical & mathematical form.

Eg- It is Sunday

Syntax of Propositional logic:-

- (A) Atomic Propositions:- It consists of single proposition symbol.
- (B) Compound Proposition:- It is constructed by combining simple or atomic propositions using logical connectives

Logical connectives:-

- | | | | |
|-------------------|--------------------|-------------------|-----------------------|
| (i) Negation | \neg | (iv) Implication | $P \rightarrow Q$ |
| (ii) Conjunction | $P \wedge Q$ (AND) | (v) Biconditional | $P \leftrightarrow Q$ |
| (iii) Disjunction | $P \vee Q$ (OR) | | |

Precedence of connectives:-

- | | |
|----------------------------|---------------------------------|
| 1 st precedence | Parenthesis () |
| 2 nd | Negation \neg |
| 3 rd | AND \wedge |
| 4 th | OR \vee |
| 5 th | Implication \rightarrow |
| 6 th | Biconditional \leftrightarrow |

Logical Equivalence:-

$\neg A \vee B$ & $A \rightarrow B$
column

Properties of Operators:-

Commutativity:- $P \wedge Q = Q \wedge P$ or $P \vee Q = Q \vee P$

Associativity:- $(P \wedge Q) \wedge R = P \wedge (Q \wedge R)$

$(P \vee Q) \vee R = P \vee (Q \vee R)$

Identity:- $P \wedge \text{True} = P$, $P \vee \text{True} = \text{True}$

Distributive:- $P \wedge (Q \vee R) = (P \wedge Q) \vee (P \wedge R)$

$P \vee (Q \wedge R) = (P \vee Q) \wedge (P \vee R)$

De Morgan's Law:-

$\neg(P \wedge Q) = \neg P \vee \neg Q$

$\neg(P \vee Q) = \neg P \wedge \neg Q$

Double negation:- $\neg(\neg P) = P$

Limitations of Propositional Logic:-

We can't represent relations like ALL, some or none.

Eg - (a) All the girls are intelligent.

(b) Some apples are sweet.

First Order Logic \Rightarrow It is an extension to propositional logic.

FOL is a powerful language that develops information about the objects in a more easy way and can also express the relationship b/w those objects.

\rightarrow FOL assumes facts, objects, relations & function.

\rightarrow FOL also has 2 main parts: $\left\{ \begin{array}{l} \rightarrow \text{Syntax} \\ \rightarrow \text{Semantics} \end{array} \right.$

Syntax of FOL \Rightarrow

Constant \rightarrow 1, 2, A, ...

Variables \rightarrow x, y, z, a, b, ...

Predicates \rightarrow Brother, father, ...

Function \rightarrow sqrt, ...

Connectives \rightarrow $\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow$

Equality \rightarrow $=$

Quantifier \rightarrow \forall, \exists

Atomic Sentences:- These sentences are formed from a predicate symbol followed by parenthesis.

Predicate ($t_1, t_2 \dots t_n$)
Eg - R & A are brothers \Rightarrow Brothers(R, A).

Complex Sentences:- These are made by combining atomic sentences using connectives.

x is an integer
Subject Predicate

Quantifiers in FOL:-

- (A) Universal (for all, everyone, everything) \forall
(B) Existential (for some, at least one) \exists

$\forall x \text{ man}(x) \rightarrow \text{drink}(x, \text{coffee})$

$\exists x : \text{boys}(x) \Rightarrow \text{intelligent}(x)$

Properties of Quantifiers:-

$$\forall x \forall y = \forall y \forall x \quad \checkmark$$

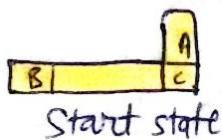
$$\exists x \exists y = \exists y \exists x \quad \checkmark$$

$$\exists x \forall y \neq \forall y \exists x \quad \checkmark$$

Planning in AI \Rightarrow

It is about the decision making tasks performed by robots or computer programs to achieve a specific goal.

The execution of planning is about choosing a sequence of actions a high likelihood to complete the specific task.



Goal Stack planning : - It is specifically used by STRIPS.

Stack is used to hold action & satisfy the

goal. A knowledge Base is used to hold current state, action.

→ Goal stack is similar to a node in a search tree where the branches are created if there is a choice of an action.

Non Linear planning : - This planning is used to set a goal stack and is included in search space of all possible subgoal orderings.

Probabilistic Reasoning in AI →

Uncertainty → Suppose if A is true then B is true means

$A \rightarrow B$. In this situation we can certain.

But if we are not sure about whether A is true or not then that situation is called uncertainty.

Causes of Uncertainty →

- (i) Information occurred from unreliable sources.
- (ii) Experimental errors
- (iii) Equipment fault
- (iv) Temp. variation
- (v) Climate change.

Probabilistic Reasoning → It is a way of knowledge representation where we apply concept of probability to indicate the uncertainty in knowledge. We combine probability theory with logic to handle uncertainty.

2 ways to solve problems :-

- Bayes' Rule ✓
- Bayesian Statistics ✓

Probability :-

Uncertainty → $0 \leq P(A) \leq 1$ → certainty

$$P(\neg A) + P(A) = 1$$

Conditional probability :-

$$P(A/B) = \frac{P(A \cap B)}{P(B)}$$

$$P(A|B) = \frac{P(A \cap B)}{P(B)}, \quad P(B|A) = \frac{P(A \cap B)}{P(A)}$$

Bayes' Theorem \rightarrow It determines the probability of an event with uncertain knowledge.

\rightarrow It is a way to calculate the value of $P(B|A)$ with knowledge of $P(A|B)$

\rightarrow It allows updating probability prediction of an event by observing new information of the real world.

$$P(A \cap B) = P(A|B) P(B) \quad \text{or} \quad P(A \cap B) = P(B|A) P(A)$$

$$P(A|B) = \frac{P(B|A) P(A)}{P(B)} \quad \rightarrow \text{It is a Bayes' Rule.}$$

It shows simple relationship b/w joint & conditional probabilities. $P(A|B)$ is posterior

$$P(A_i|B) = \frac{P(A_i) P(B|A_i)}{\sum_{i=1}^K P(A_i) \cdot P(B|A_i)} \quad \left[\text{In general } P(B) = P(B|A) P(A) \right]$$

Q. What is the probability that a patient has diseases meningitis with a stiff neck?

A doctor is aware that disease meningitis causes a patient to have a stiff neck & it occurs 80% of time.

$$P(D) = 1/30000, \quad P(S) = 2\%$$

Aj-

$$P(S|D) = 0.8$$

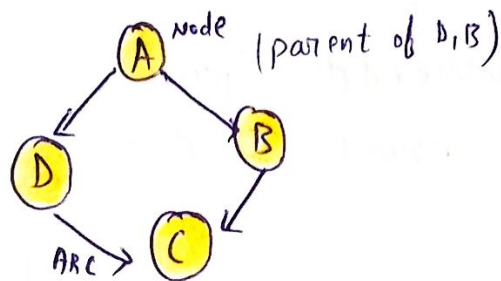
$$P(S) = 0.02, \quad P(D) = \frac{1}{30000}$$

$$P(D|S) = \frac{P(S|D) P(D)}{P(S)} = \frac{0.8 \times \frac{1}{30000}}{0.02} = 0.00133$$

Bayesian Network \Rightarrow It is a probabilistic graphical model which represents a set of variables and their conditional dependencies using a directed acyclic graph. [DAG]

It can be used for building models from data & experts opinions and it consists of 2 parts:-

- Directed Acyclic Graph
- Table of conditional probabilities



It has mainly 2 components:
 → Causal component
 → Actual Numbers

To propagate belief in Bayesian Nlw initial "Directed Acyclic Graph" is converted into an undirected graph in which the arcs can be used to transmit probabilities in dirⁿ of evidence

T	0.2
F	0.8

(A)

T	0.1
F	0.9

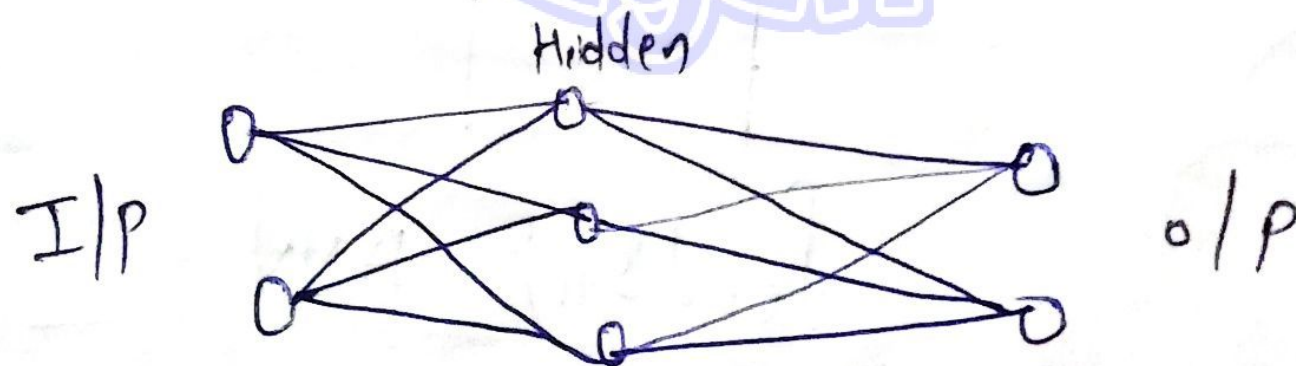
(B)

(C)

A	B	$P(C=T)$	$P(C=F)$
T	T	—	—
T	F	—	—
F	T	—	—
F	F	—	—

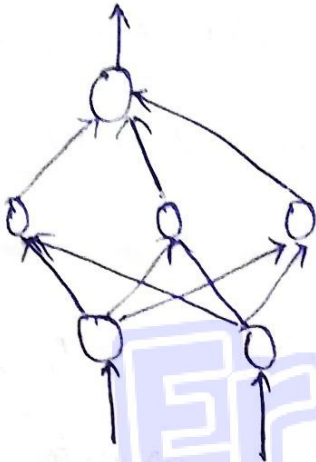
(a) Neural Networks :- These are computing systems inspired by the biological neural networks that constitute animal brains.

The idea of ANNs is based on belief that working of human brain, by making right connections, can be imitated using silicon & wires as living neurons & dendrites.



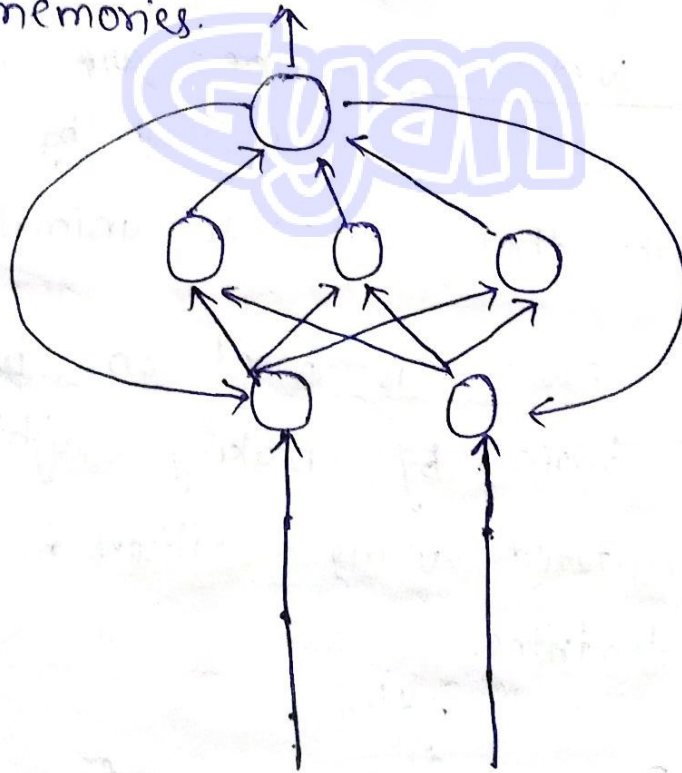
Types of ANN:—

- (i) Feed Forward ANN: \Rightarrow The information flow is unidirectional. A unit sends information to other unit from which it does not receive any information. There are no feedback loops. (4)



Feedforward ANN

- (ii) Feed Back ANN: \Rightarrow Feedback loops are allowed. They are used in content addressable memories.



Feed Back ANN