AIDS-1

→ An agent is anything that can be viewed as perceiving its environment through sensor, acting upon that environment through actuators.

→ A software agent recieve keystroke, file contents, and network packet as sensory inputs and acts on environment by displaying on screen, writing files, and sending network packets.

Agent terminology:-

- performance measure of agent
- Behaviour / action of agent - action performed, after specified sequence of percepts
- Percept - perceptual inputs at specified instance.
- Percept sequence - history of everything that agent has perceived till date.
- Agent function - map from percept sequence to action

a 2 F(P)

- Fully Observable / partially Observable

→ If an agent's sensor gives it access to complete state of environment at each point in time, then task envi. is fully observable.

→ An env. might be partially observable, if sensor contain noisy data or inaccurate sensors.

Single Agent:-

In a SA environment, there is only one entity or agent responsible for performing task within the environment.

Ex:- Robot Assigned to clean the room.

Multiple Agent:-

In MA environment, there are multiple agents, each with its own goal, and they may interact with or influence each other.

Ex:- Two robots assigned to clean different rooms.

Page No.
Date

Static environment:-
the elements and features of environment do not change over time.

Dynamic environment:-
the environment and features change over time.

Deterministic Environment:-
The outcome of an action is entirely predictable, given current state of env. and action taken.

Stochastic
The outcome of an action has some level of randomness or uncertainty, even if current state and action are same.

Discrete:-
set of possible states and actions is finite, countable or distinct.

Continous:-
set of possible states and action is uncountable and can take on continous range of values.

Episodic:-
The agent's experience is divided into episodes, and each episode is a self contained task with clear beginning and end. The agent action with one episode do not affect subsequent episodes.

Sequential:-
The agent's action has lasting impact on future states, and current state depends on agent's history of actions and observables.

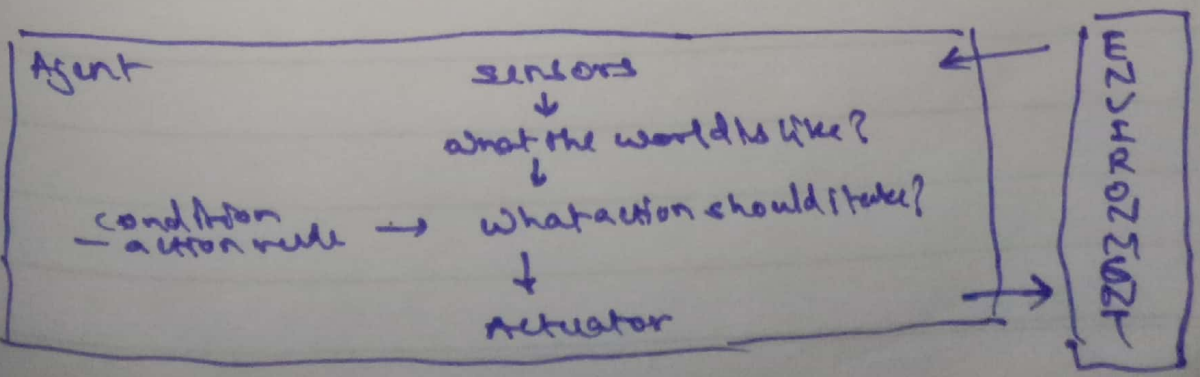
PEAS Description:-

performance
Environment
Actuators
Sensors.

Types of Agents:-

i) Simple-Reflex Agent :-
Selects an action based on the current percept ignoring percept history. · If {set of percepts} then {set of actions}

Limitations:-
* Sensors:- Collect info about current state.
Rules:- Define rules based on input
Actuators :- execute actions.



2. Model based Agent :-

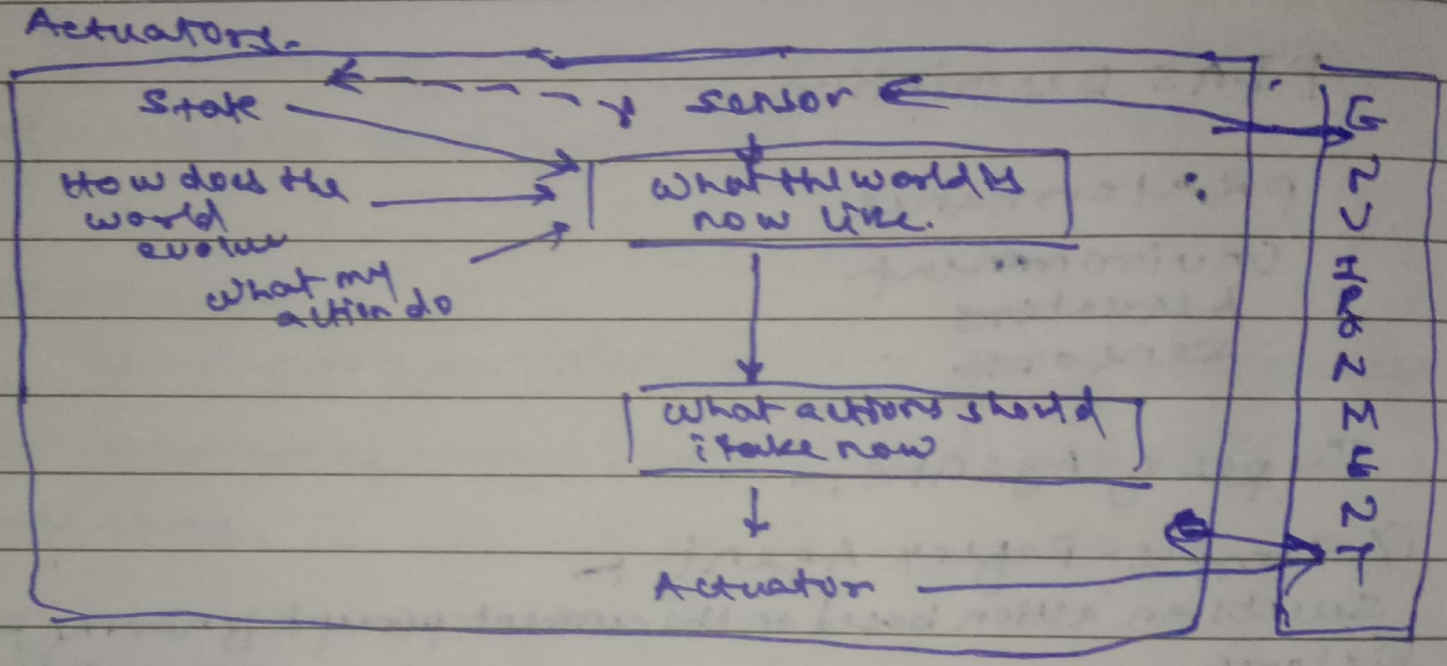
→ Also, takes into consideration perception history.

→ The internal model helps take more informed decision by anticipating consequence of action.

percept History.

Update Model

Decision Making



Goal Based :-

This agent operates on considering the goal and making decisions to achieve them.

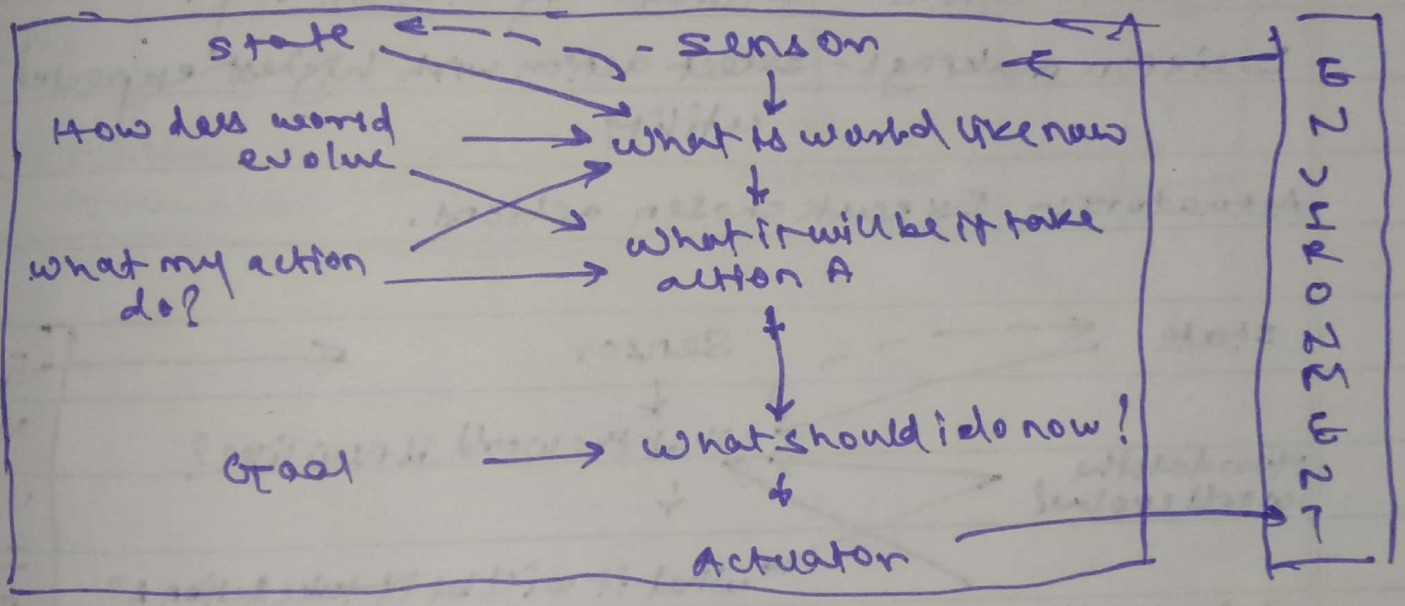
This operates on achieving something specific rather than just reacting to environment.

Goal :- Define Goals or objectives the agent wants to achieve.

Action Planning :- Develop a plan or sequence of action.

Decision Making :- Make decisions based on the current state.

Actuators :- Make actions based on decision made.



Page No.
Date

Utility Based

→ makes decision by considering utility or desirability of multiple action.

→ the agent evaluates the outcomes of various action and selects one with highest expected utility.

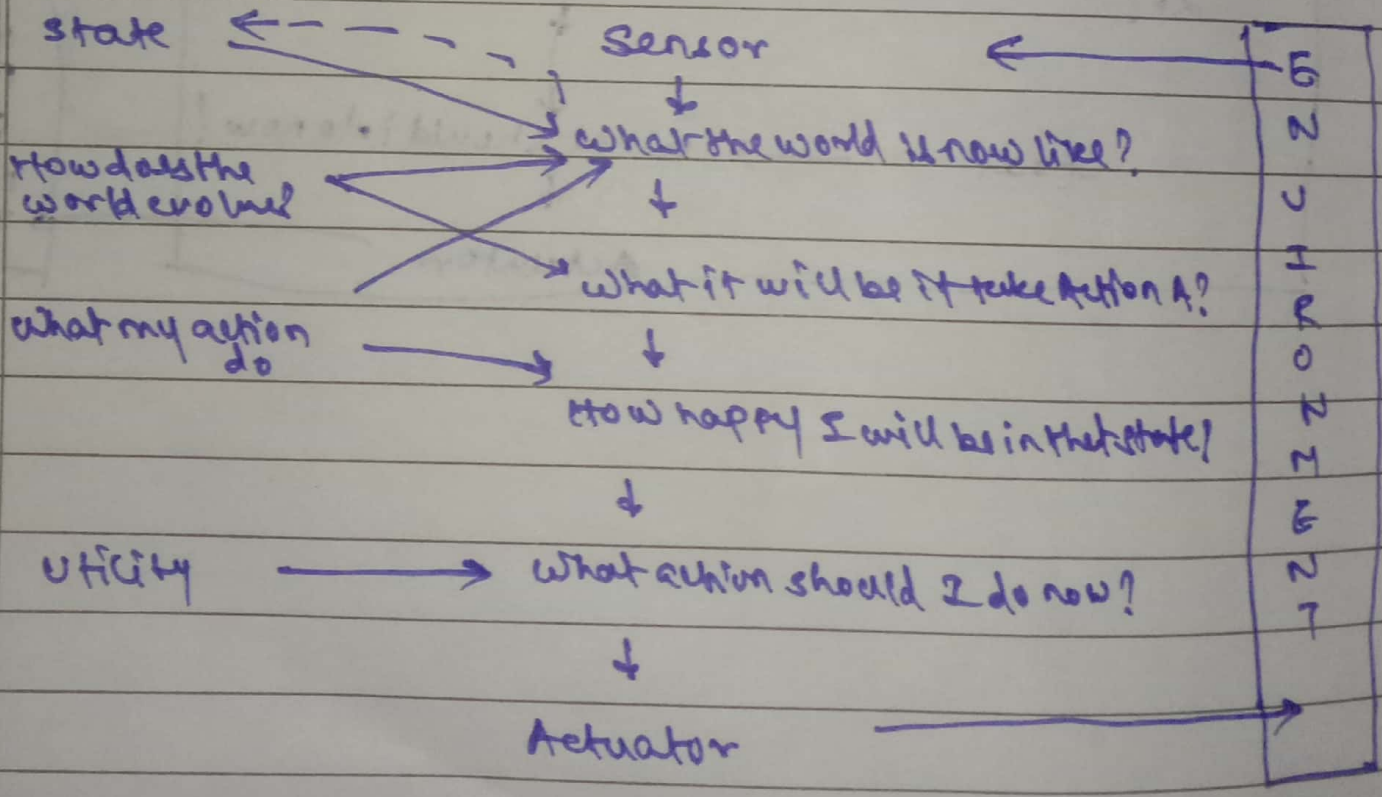
Define utility :-

set utility function that assigns a numerical value to each possible state.

Evaluate action :- estimate expected utility.

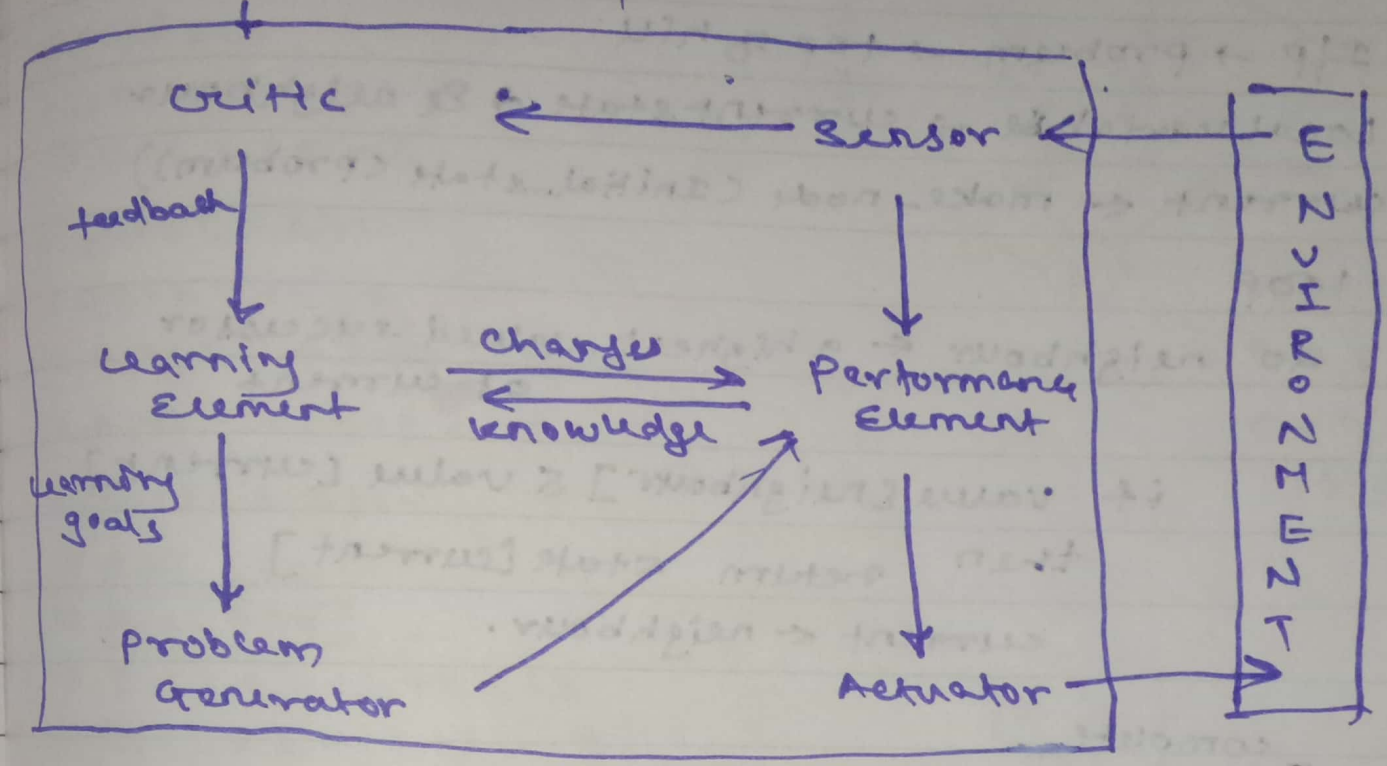
Decision making :- select action with highest expected utility.

Actuators :- Execute chosen actions.



Learning Agent:-

By actively exploring and experimenting with their env.



Page No.
Date

Hill climbing Algorithm :-

I/p → problem → top of hill

Local variable → current state & neighbour

current ← make_node (Initial_state (problem))

loop

do neighbour ← a highest_valued successor of current

if value [neighbour] ≤ value [current]

then return state [current]

current ← neighbour.

not ↗ complete

↘ optimal

Limitations :-

i) Local_maxima

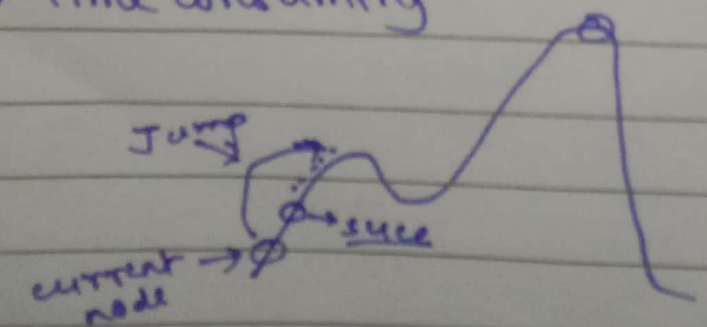
ii) Ridge

iii) Plateau

Steepest Ascent Hill climbing

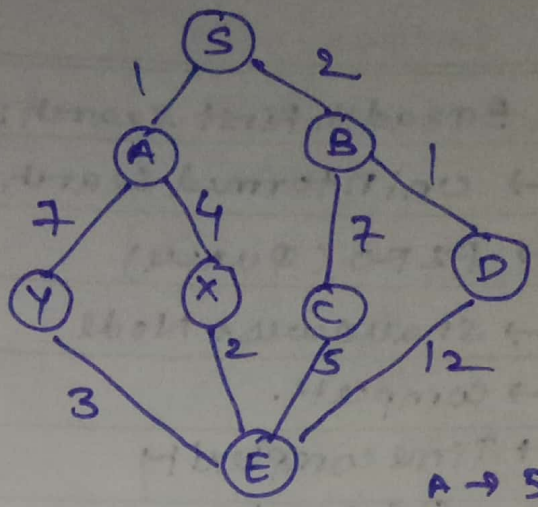
→ Optimal solution

→ Time Consuming



A* Algorithm

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



A → 5 , X → 5
B → 6 Y → 8
C → 4
D → 15

Expand S :-

(S, A) = 1 + 5 = 6
(S, B) = 2 + 6 = 8

Expand (S, A) :-

(S, B) = 2 + 6 = 8
(S, A, Y) = 1 + 7 + 8 = 16
(S, A, X) = 1 + 4 + 5 = 10

Expand (S, B) :-

(S, A, Y) = 16
(S, A, X) = 10
(S, B, C) = 2 + 7 + 4 = 13
(S, B, D) = 2 + 1 + 15 = 18

Expand (S, A, X) :-

(S, A, X, E) = 1 + 4 + 2 = 7

Completeness - Yes
Optimality - Yes

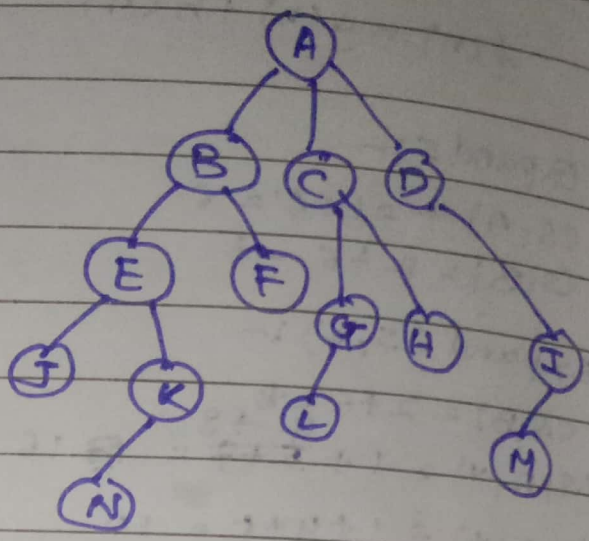
Time complexity - $O(b^m)$
Space complexity - $O(b^m)$

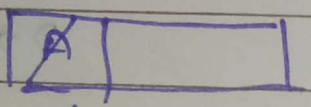
branching factor = b
depth = m

Date

Breadth First Search :-

- → Uninformed search
- → FIFO (Queue)
- → Shallowest Node
- → Complete.
- → Time complexity
  - $O(V+E)$
  - $O(b^d)$ → branch factor
    d → depth.





→ ~~B~~ C D

→ ~~C~~ D E F

→ ~~D~~ E F G H

→ ~~E~~ F G H I

→ ~~F~~ G H I J K

→ ~~G~~ H I J K

→ ~~H~~ I J K L

→ ~~I~~ J K L

→ ~~J~~ K L M

→ ~~K~~ L M

→ ~~L~~ M N

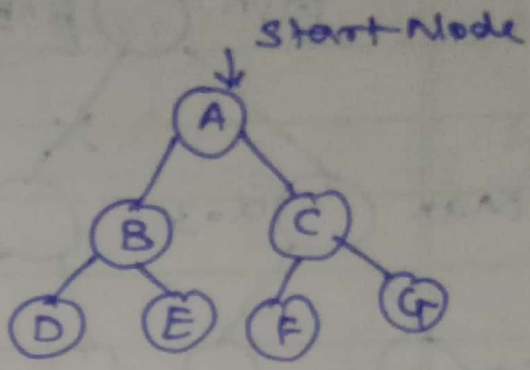
→ ~~M~~ N

→ N

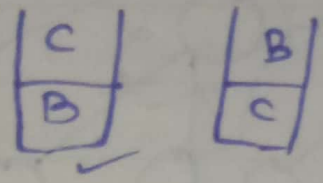
Depth First Search :-
- Uninformed search technique
- Stack (LIFO)
- Deepest Node
- Incomplete.

Non-Optimal

Time complexity :
$O(b^d)$.

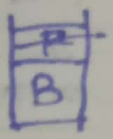


A

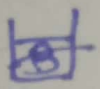


AC

ACG



ACGF



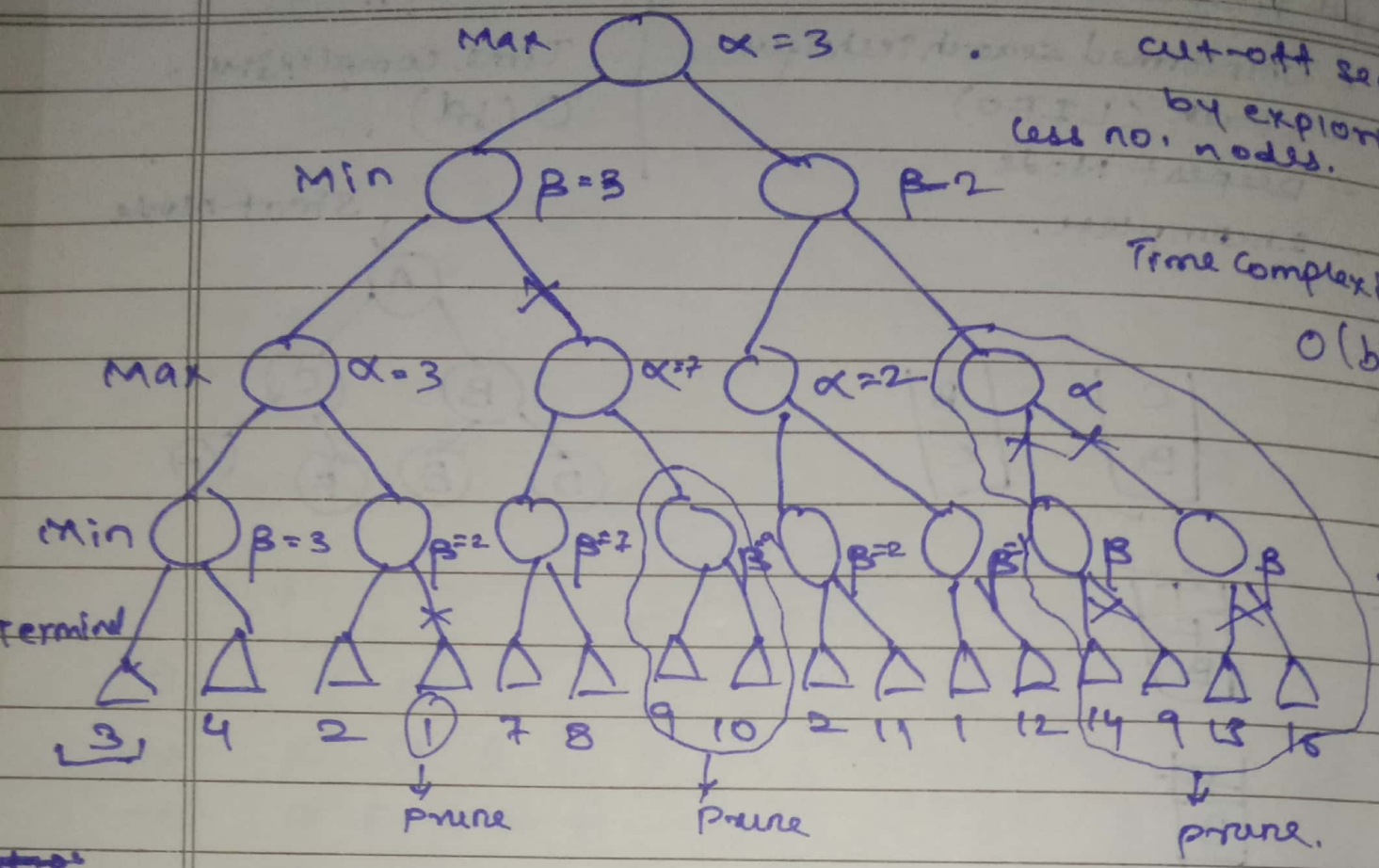
ACGFB

ACGFBE

ACGFBED.

α-β pruning

Page No.
Date

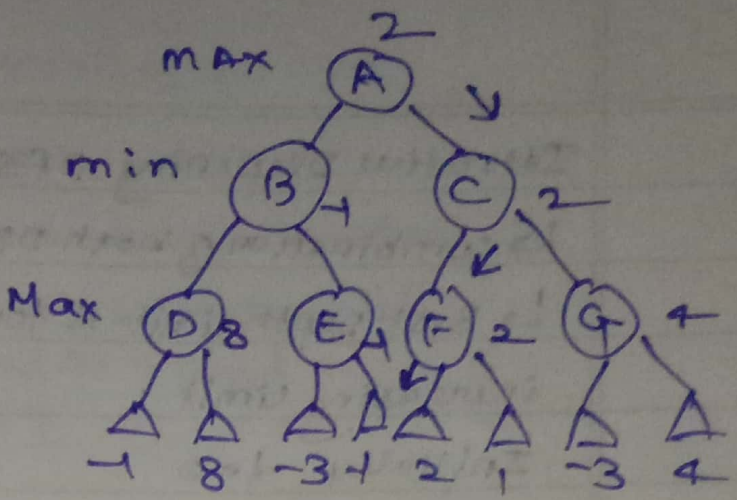


cut-off search by exploring less no. nodes.

Time Complexity $O(b^d)$

MiniMax Algorithm

- Backtracking Algorithm
- Best move strategy used
- Max will try to maximize its utility. (Best)
- Min will try to minize. (worst move)
- Time complexity $O(b^d)$

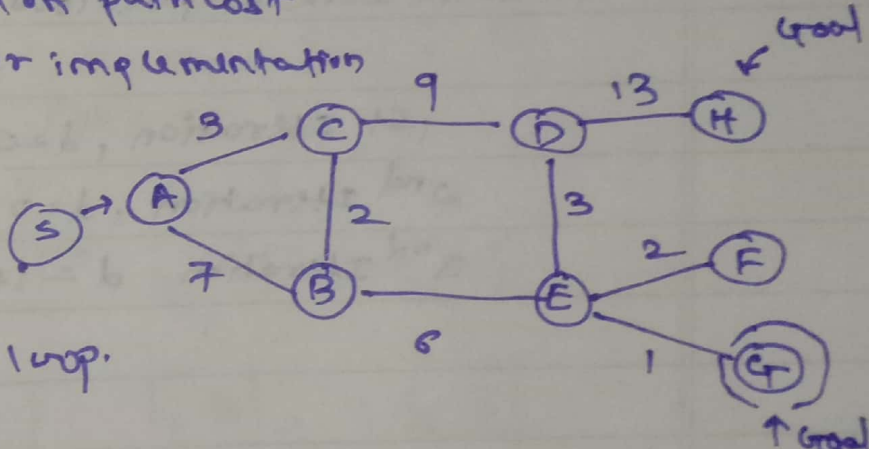


Uniform cost search Algorithm:- {Backtracking}

- Used for weighted Tree / Graph Traversal
- Goal is to find the optimal path
- Node expansion is based on path cost
- Priority queue used for implementation

Advantage :-
- optimal solution

Disadvantage
- Struck in infinite loop.



∴ Path to Goal Node G :-

A → C → B → E → G

(12)

∴ Path to Goal Node H

A → C → B → E → G → F → D → H

(33)

Page No.
Date

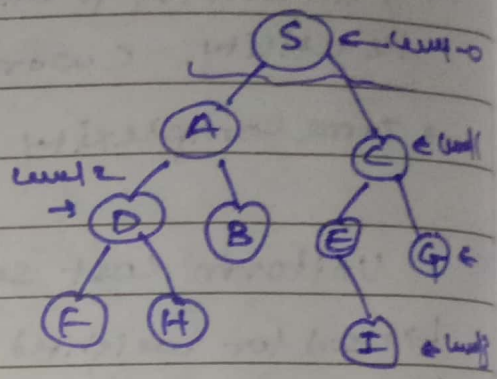
Iterative Deepening DFS

↳ combination of both DFS and BFS

↳ Best Depth limit is found out by gradually increasing limit.

Initially d=0

↳ every iteration increase by 1.



Advantage:-

Includes benefits of both BFS and DFS

fast searching and less memory

Disadvantage:-

Repeats the work at previous phase.

1st Iteration, d=0 [S]

2nd Iteration, d=0+1 [S → A → C]

3rd Iteration d = 1+1 [S → A → D → B → C → E → G]

Crypt Arithmetic Problem :-

constraints :→ No two letters have same value
↳ sum of digits must be as shown in problem
↳ There should be only one carry forward.
↳ Digits can be assigned to a word
[0-9]

```
  TO
+ GO
------
 OUT
```

O = 1
∴ T = 2
Let G = 8
∴ U = 0

```
  SEND
+ MORE
-------
 MONEY
```

M = 1
S + M ≥ 10
∴ S = 9
O = 0

C4 C3 = 1 C2 = 1 C1 = 1

S 9	E 5	5	N 6	6	D 7
M 1	O 0		R 8	9	E 5
M 1	O 0		N 6	6	E 5	5	Y 2

∴ $C_2 = 1$
Let E = 5
∴ N = 6

Let R = 9
but S ≠ 9
Now
Let R = 8

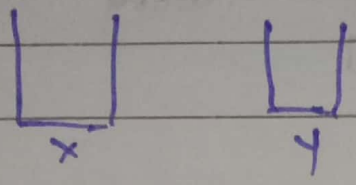
Start with leftmost.

Rem:- Graph Coloring
Simulated Annealing
Bidirectional.

Page No.
Date

Water Jug Problem and its State Representation :-

Two jugs of different capacities are given.



↳ Goal is to fill exactly '2' litres of water into 'y'-litre jug

State is represented as $\langle x, y \rangle$ — y: int. amount of water in y litre of Jug.

x: int. amount of water in 'x' litre Jug

GOAL:- To get exactly '1' litre of water in a 2 litre Jug.

$\langle 1, y \rangle$ → Goal state.

Two Jugs: 2L, 3L

$\langle 0,0 \rangle$ ← Initial state.

↓

$\langle 0,3 \rangle$

↓

$\langle 2,1 \rangle$

↓

$\langle 0,1 \rangle$

↓

$\langle 1,0 \rangle$

Depth Limited Search Algo.

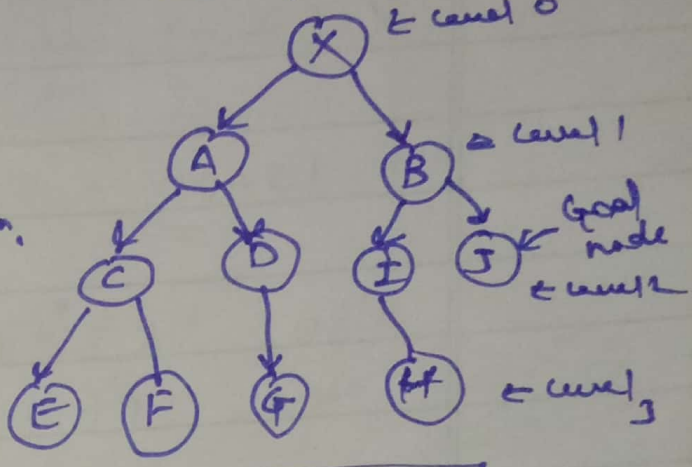
↳ working is similar to DFS but with predetermined limit.

↳ Helps in solving the problem of DFS: Infinite path.

Termination → (i) Failure value

→ (ii) Terminates on reaching predetermined depth.

Example:-



Depth, d = 2 for Goal J

X → A → C → D → B → I → J

For Goal H

This depth will result in no soln.

Advantage:-

↳ Memory efficient

Disadvantages:- → Incompleteness

↳ can be terminated without finding soln.

↳ not optimal.

↳ Time complexity: $O(b^d)$

space $O(b \times d)$

Page No.
Date

Knowledge Representation and Reasoning.

- Logic → Propositional Logic
- Logic → Predicate Logic
- Rules → if-then
- Semantic Net → Google Graph
- Frames → Slots and Fillers
- Script.

FOP, FOPL
Resolution
FWD chaining
BWD chaining

Representing simple facts in FOPL:-

'∧'	conjuction (And)
'∨'	disjunction (OR)
'→'	implication
'≡'	equivalence
$\forall_x$	for all x, such that
$\exists$	for some x, such that

(1) All Boys like Football

$\forall_x$: Boys(x) → like(x, Football)

(2) Some Boys like Football.

$\exists x$: Boys(x) ∧ like(x, Football).

Forward Chaining

It moves forward from start to goal. Also called Data Driven.

↳ Search Tree has initial configuration

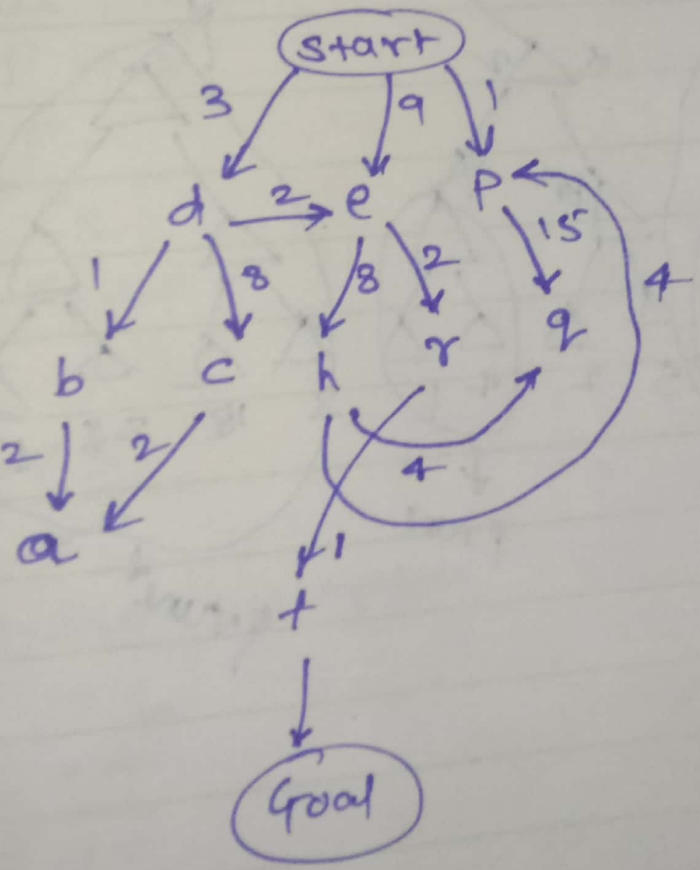
↳ Next level of Tree is generated by finding all the rules whose left side matches the root node.

Backward chaining

It moves backward from goal to start.

↳ Search tree has Goal configuration.

↳ Next level of tree is generated by finding all the rules whose right side matches root nodes.

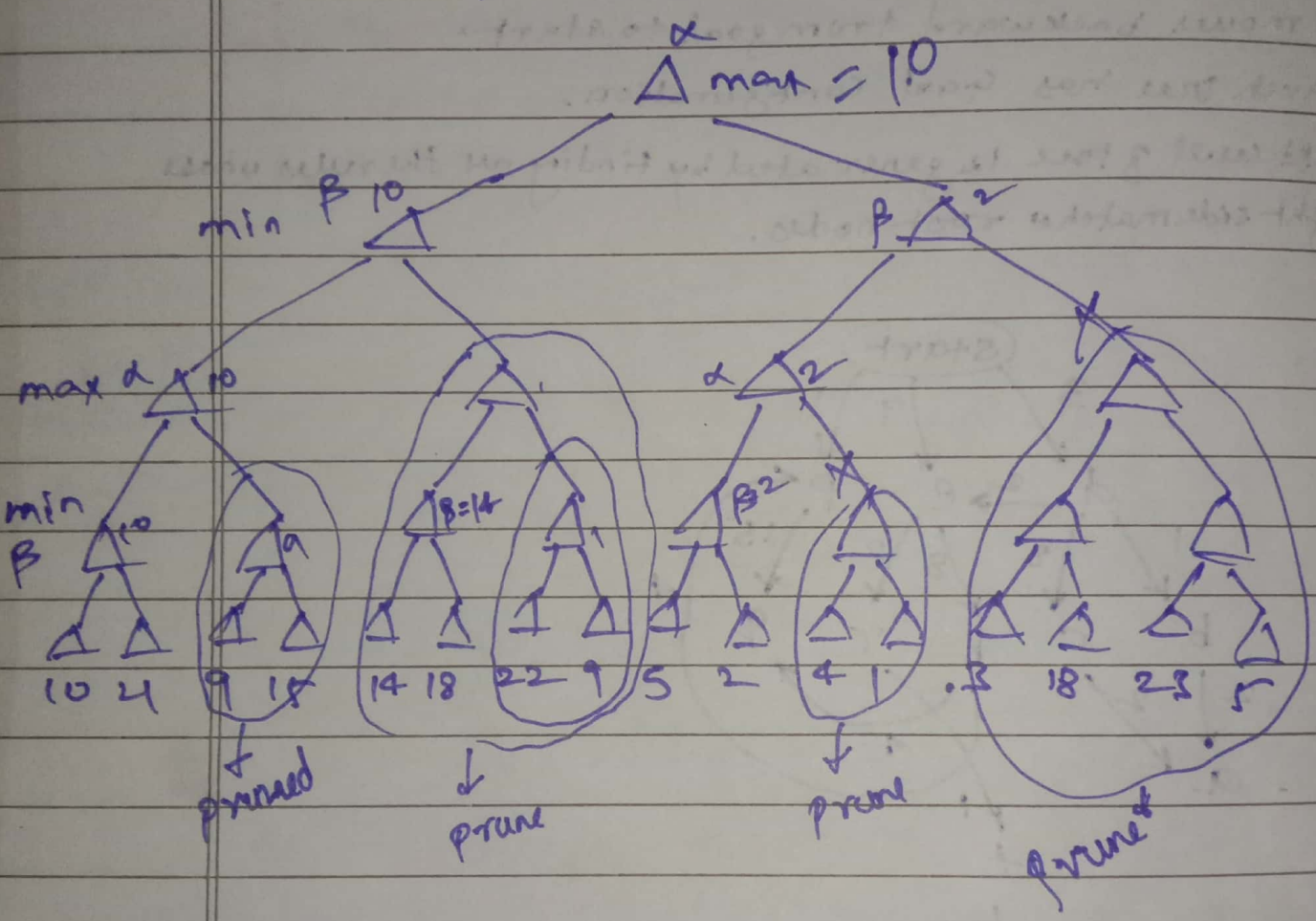


John has atleast two friends

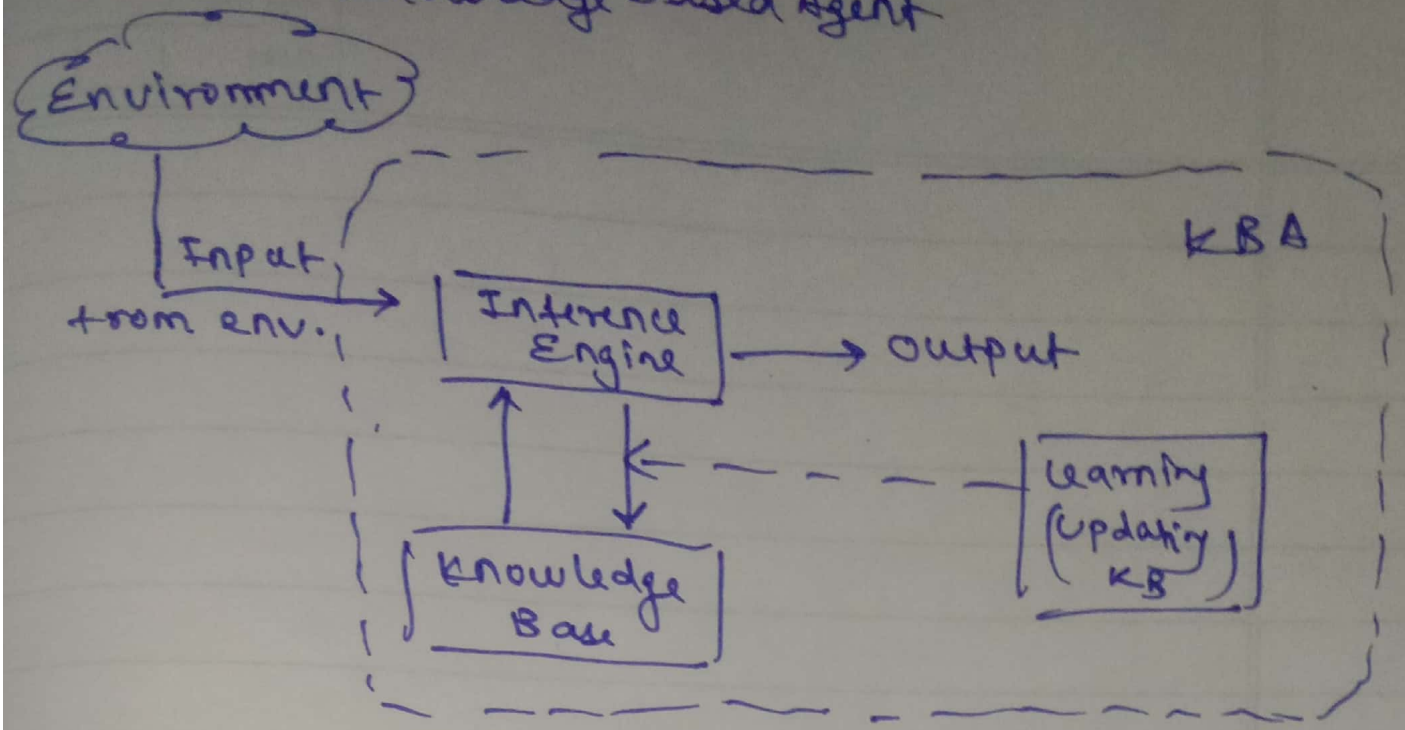
$\exists y \exists z (Friends(John, y) \wedge Friends(John, z) \wedge y \neq z)$

If two people are friends then they are not enemies

$\forall x \forall y (Friends(x, y) \rightarrow \neg Enemy(x, y))$



Knowledge Based Agent



Modus ponnem

$$\frac{p \to q \quad p}{q}$$

Modus Tollen

$$\frac{p \to q \quad \sim q}{\sim p}$$

Hypothetical syllogism

$$\frac{p \to q \quad q \to r}{p \to r}$$

Disjunctive syllogism

$$\frac{p \vee q \quad \sim p}{q}$$

Resolution

$$\frac{p \vee q \quad \sim p \vee r}{q \vee r}$$