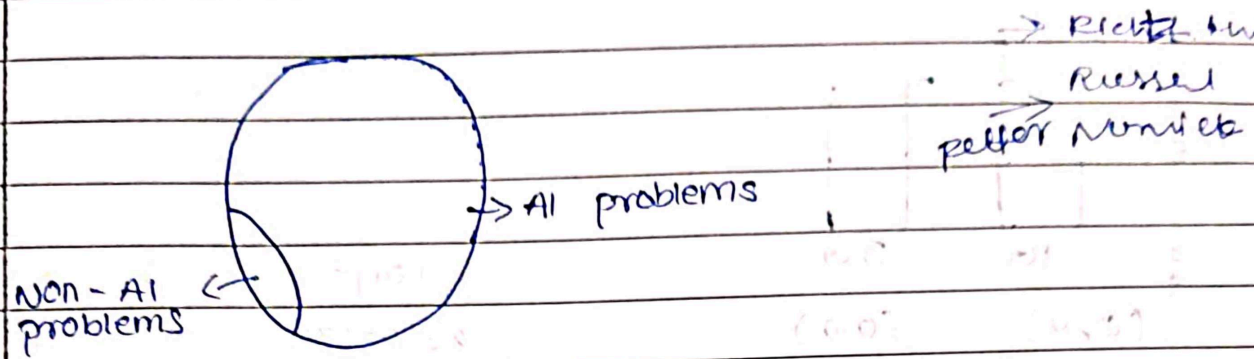
Vicky 25-24-AI

AI- based mini proj. → 14th week of sem (last week)
title approve ⇒ ↳ poster AB → theory ↳ presentation (mini proj discussion)
practical ISA ↙ ↳ presentation Date: / / ↳ theory ISA



→ Richard Knight
Russel
peter Norvig

Two types of problems

1) Structured : Also known as algorithmic problem. They have algorithmic approach. Machines are better than humans to solve structured problem.

2) ~~AI~~ NON-AI algorithmic also known as non structured problems. Humans are better than machines to solve them.

AI problems:

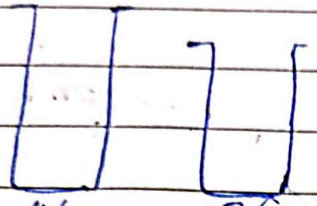
1) Mudane tasks: computer vision, NLP, hand written processing.
2) Formal tasks: game playing, proving mathematical theorem.
3) Expert tasks: medical digonesis, business pred. stock market analysis., financial analysis.

AI:

Study of making machines ~~do that~~ do what humans are better at the moment (making machines do eg NLP)

Date: / /

⇒ Water Jug problem



(0,0)	(0,0)
(4,0)	(0,3)
(1,3)	(3,0)
(1,0)	(3,3)
(0,1)	(4,2)
(4,1)	(0,2)
⇒ (2,3)	(2,0)
(2,0)	

A

12/02/2024 → SPCC & CSS
26/02/2024 → MC & AI
04/03/2024 → QA

11/03

18

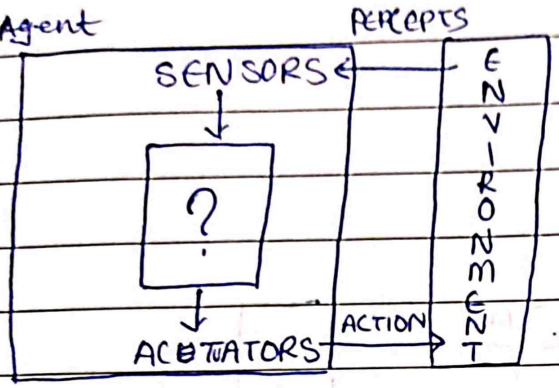
01
08
15
22

Date:

Module 2: Agent & Enviornment

Agent is system which has AI imparted into it, also called as Intelligent system.

? → thinking



Agent has two important body parts i) Sensors: Agent pursues environment through sensors.

ii) Actuators / Effectors: Agent uses actuators to perform action on environment. Agent is designed to interact with environment which has predefined goals.
Agent pursues environment through sensors, decide action to be performed and takes the action using actuators until the environment reaches the goal state.

Eg: Agent: self driving car
Environment: Roads, signal, other traffic, sign board
Sensors: Camera, proximity sensors, gps
Actuators: steering, brake, accelerator, horn, etc.
Goal: reaching destination safely, comfortably, with minimum time.

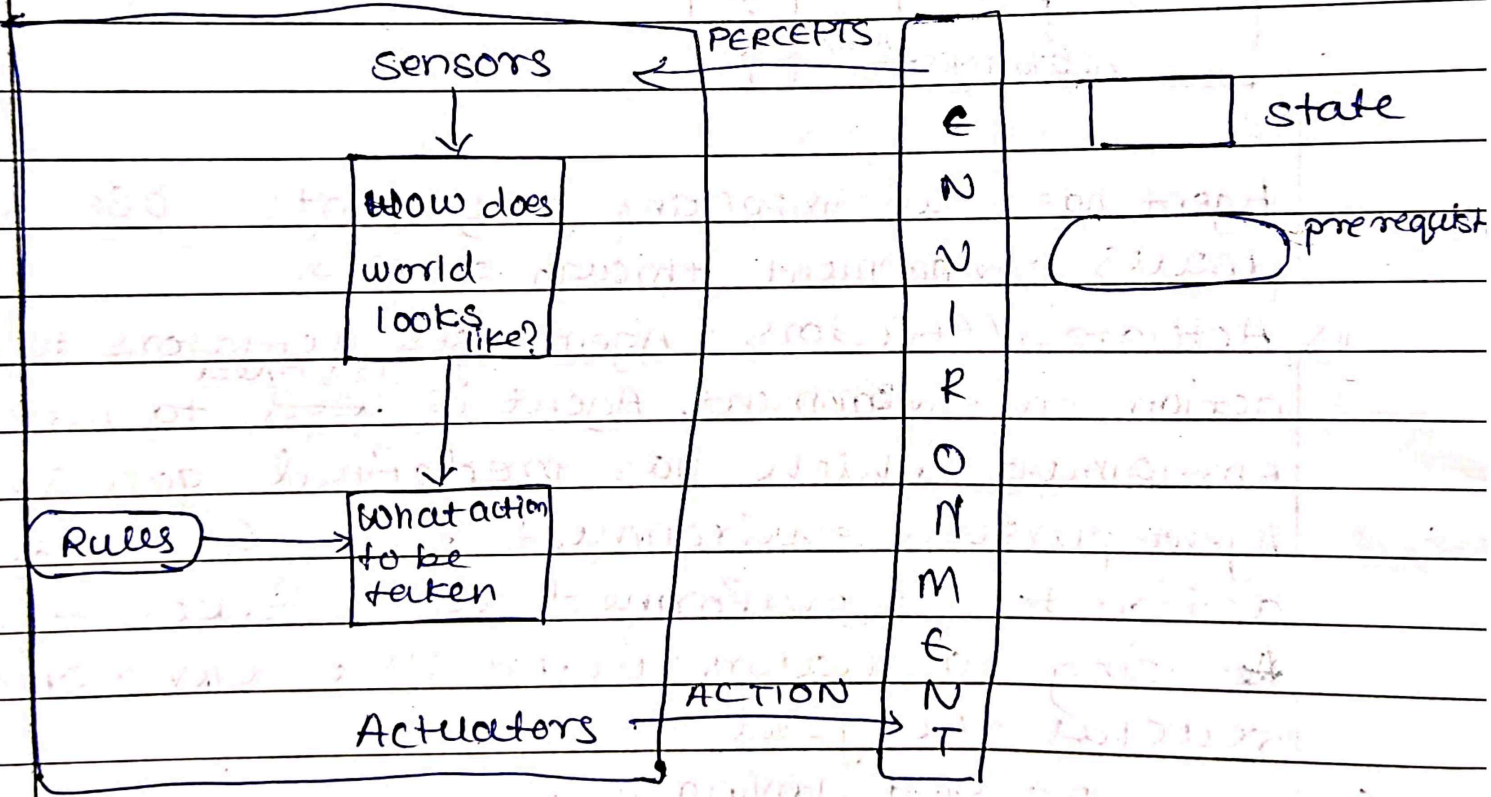
Date: / /

⇒ Types of Agents

1) Simple Reflex agent.
2) Model based agent.
3) Goal based agent
4) Utility based agent.
5) Perfect learning agent

1) Simple Reflex agent

⇒

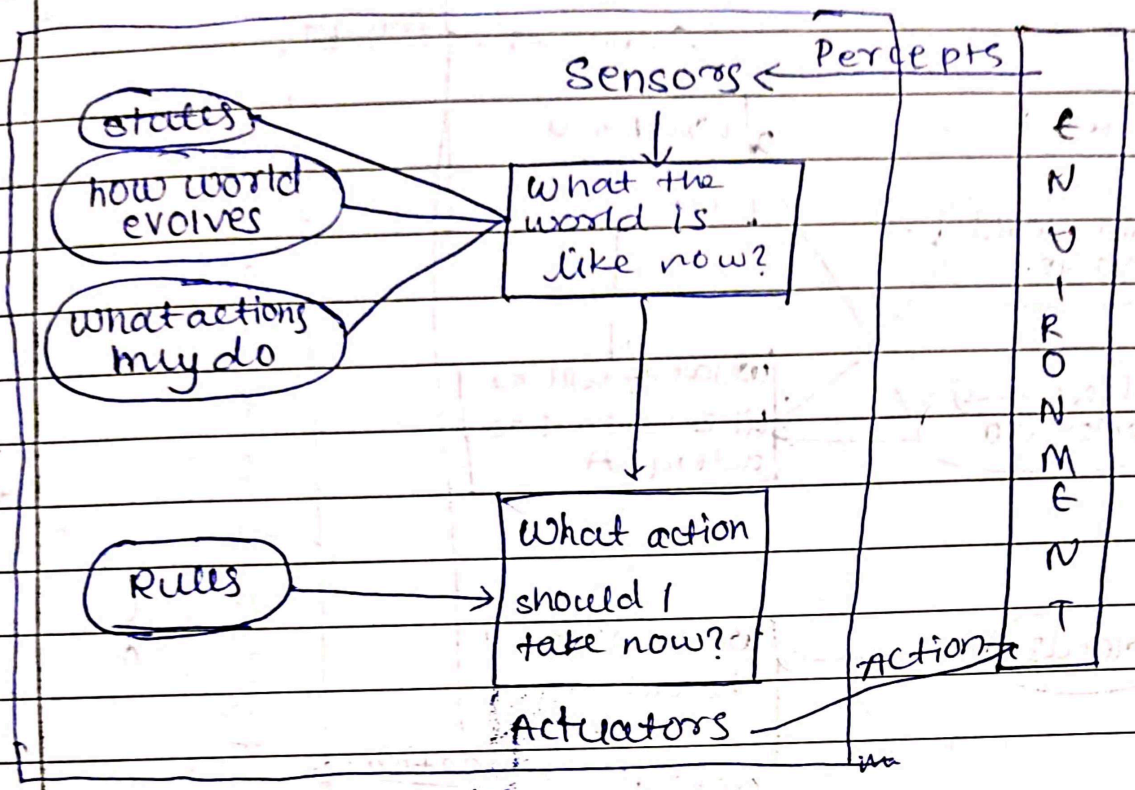


This agent is designed to interact with fully observable environment (water jug problem)

⇒ Simple reflex agent maps the current percept directly to the action without involving thinking inbetween.

Date: / /

2) Model Based Agent



→ Partially observed environment.

→ Difficult to figure the current state with incomplete percepts. For this agent has to refer model of environment which includes

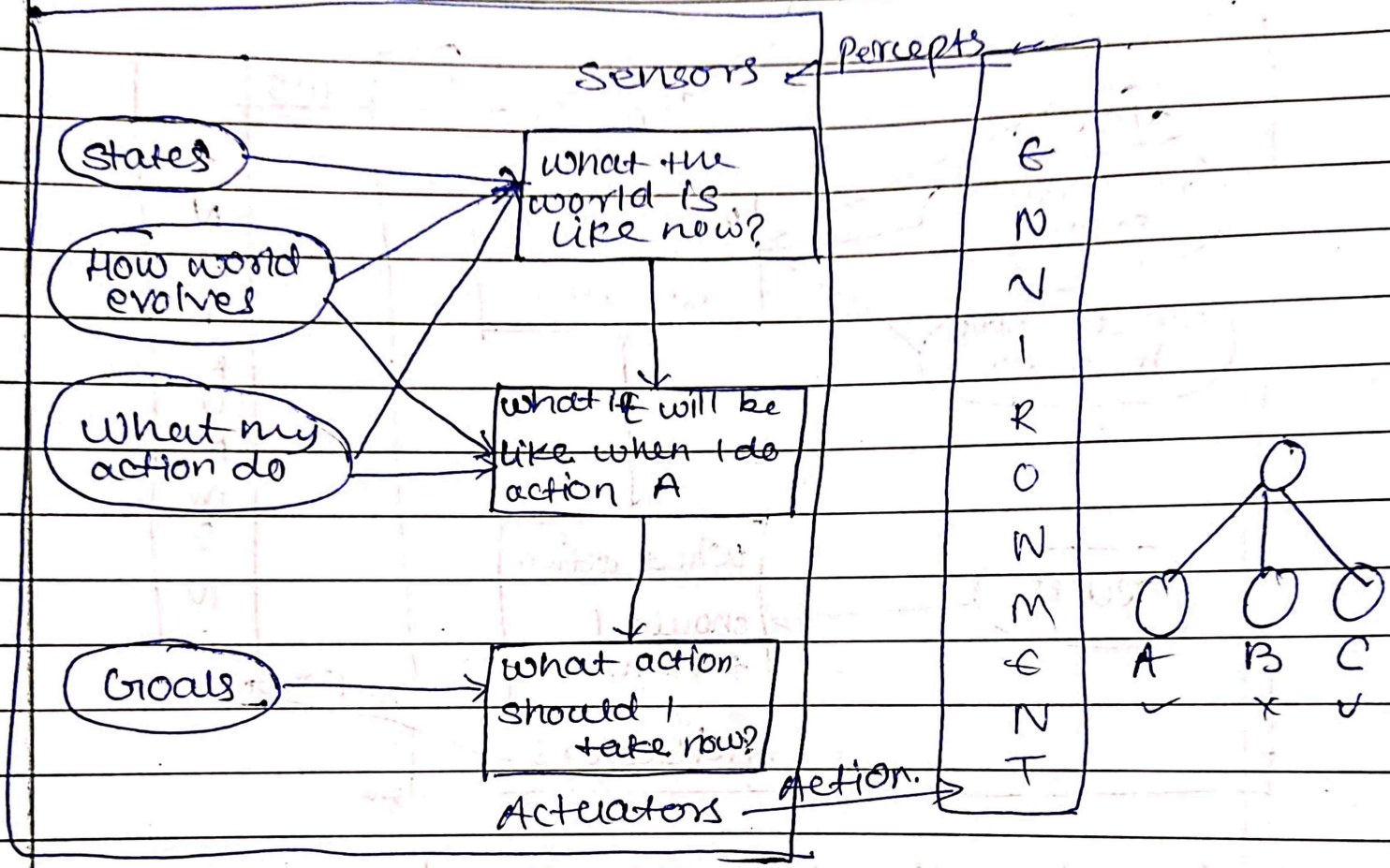
i) states, some situation related to environment with past experience.

ii) How the world evolves: This procedure tells how and why things are changing in the environment.

iii) What my action do: This procedure tells use of action

Date: / /

Goal Based Model

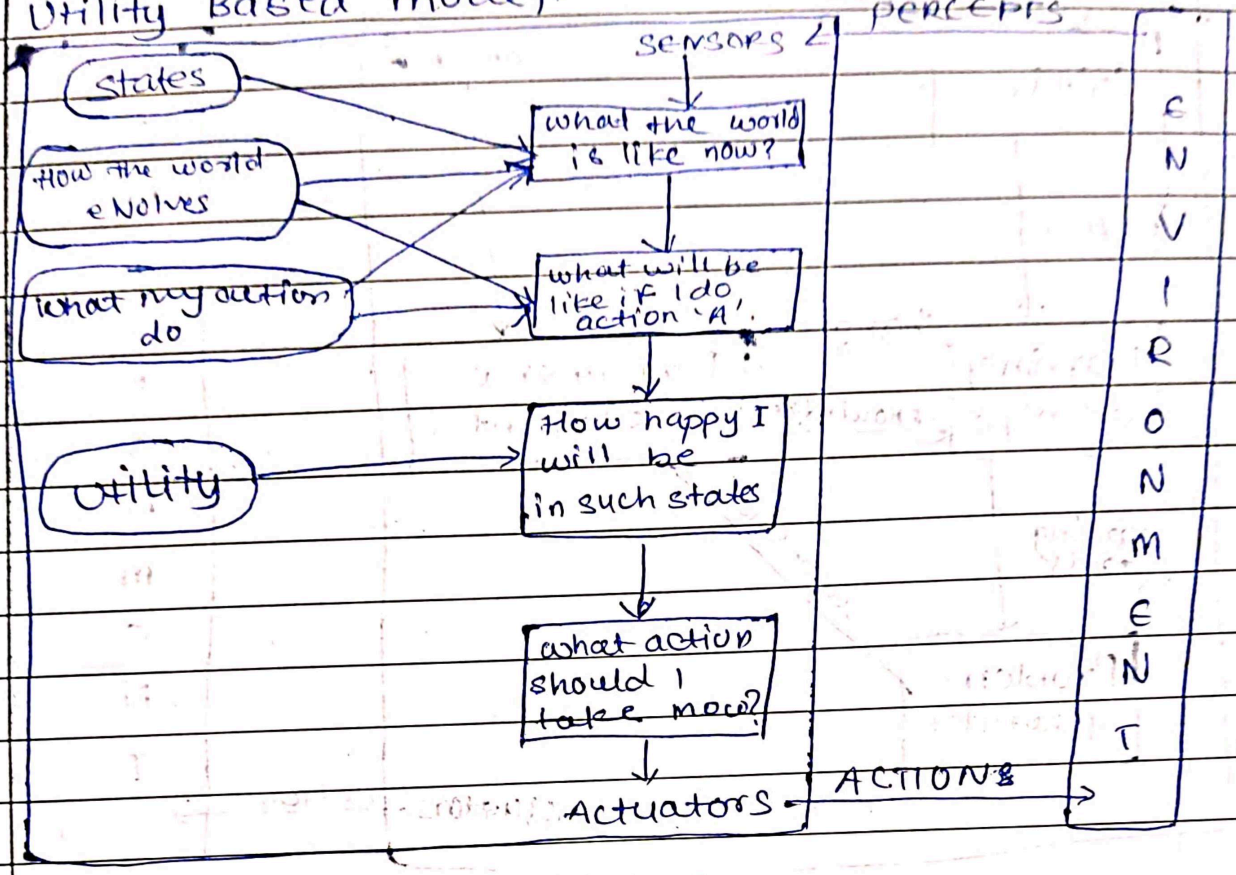


→ The goal based agent will have information (description) about goal.

→ capable of pre-determine the next state from current state. It will always choose the action which leads towards goal.

Date: / /

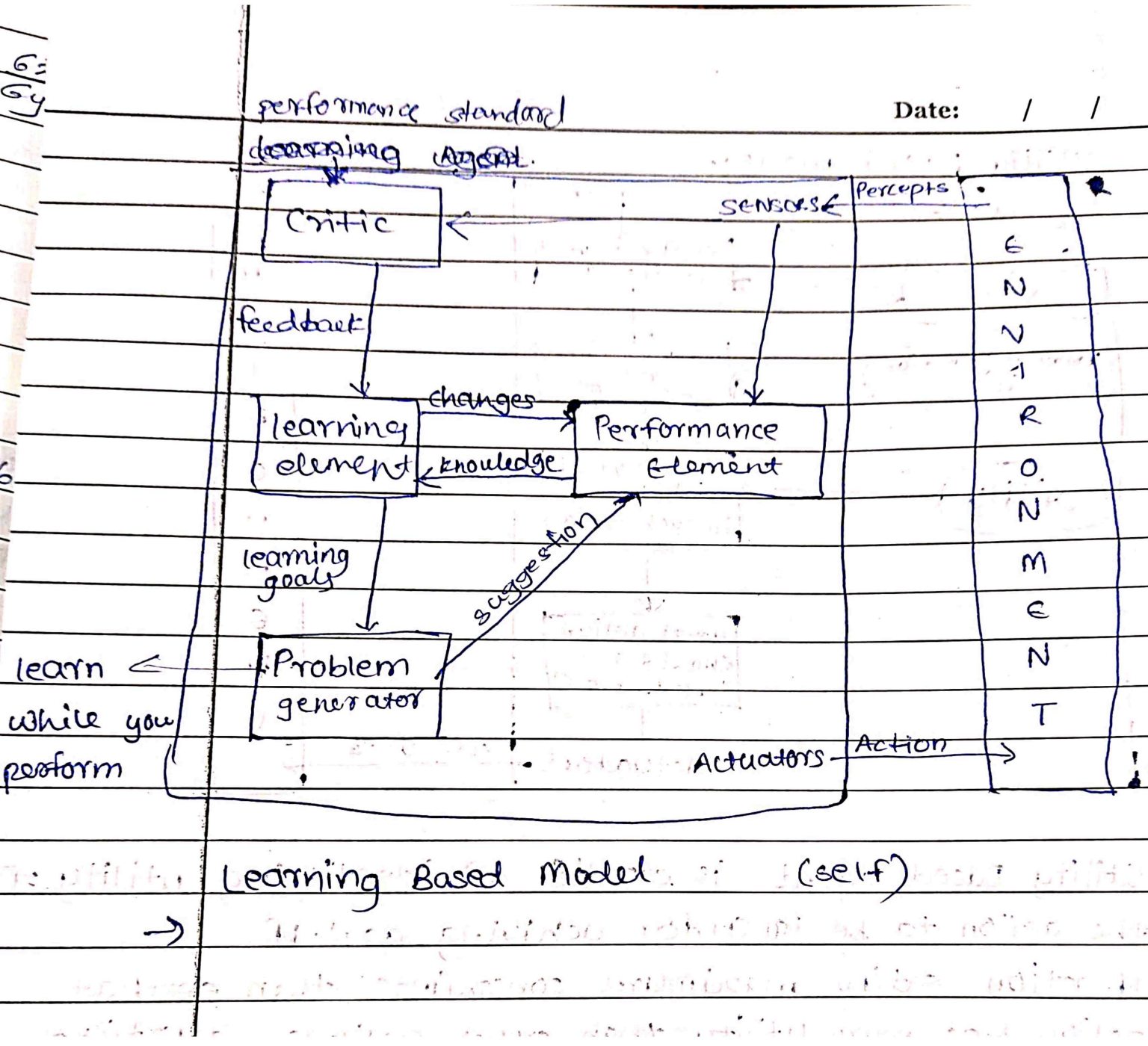
Utility Based model.



→ Utility Based agent is capable of identifying utility of the action to be taken for achiving goal. If

→ If action satify maximum constraints then that action has more utility than other actions. quotient

→ Utility function will provide the happiness for an agent for that state.



Learning Based Model (self)

→

Date: / /

Types of Environment (Task environment)
OR
Properties of environment

1) Fully observable or partially observable.

2) Static or Dynamic

3) Deterministic or Non-Deterministic.

4) Discrete or continuous

5) ~~Apt~~ Episodic or non-episodic. (sequential)

Date: 01 / 02 / 24

* PEAS:

Characteristics of Agents.

Standard stands for performance, major measure, environment, actuators & sensors.

→ Write PEAS description for self driving car.

1. Performance measure, safety, minimum time, legal, comfortable, etc.
2. Environment → sign board, traffic signal, roads, etc.
3. Actuators → steering, accelerators, brake, horn, etc
4. Sensors → GPS, proximity sensors, camera etc

PEA

MSE

Q1 MCQ – 5 mcq

Q2 2 / 3 – 5 mks.

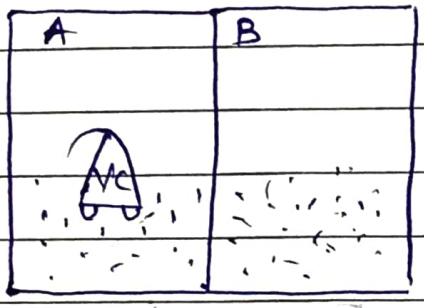
Q3 2/3 or 1/3 – 10 mks

agent & environment.

Problem Solving

PAGE NO.:

Vacuum cleaner world Problem



To adjacent rooms, vacum cleaner is in room A dirt is available in both room. the goal is to clean both the room.

1) State description:
The state can be described as the cond^n of room A and room B and also position of vacum cleaner. It can be described as ordered pair of three integer (x, y, z)
where $x \Rightarrow$ room A cond^n ($x=0$, A is clean $x=1$ A is dirty)
$y \Rightarrow$ (room B cond^n ($y=0$, B is clean $y=1$ B is dirty
$z \Rightarrow$ presence of vacum cleaner ($z=A$, $z=B$)

2) Describe initial state & goal state
Initial state = (1, 1, A) / (1, 1, B)
Goal state = (0, 0, A) / (0, 0, B)

3) List all the actions
Right: Room A to Room B left: Room B to Room A
Clean: Clean all the dirt in the room.

(1, 1, A) ← IS
→ clean A
(0, 1, A)
→ Right
(0, 1, B)
→ clean B
(0, 0, B)
→ GS

ROLL NO: 2N02A0037 CMPN A

PAGE NO.:

Assignment (Take Home Test)

Q. Solve missonary and Cannibals

In the missionaries and Cannibals problem three missionaries and three cannibals must cross a river using a boat which can carry at most two people, under the constraint that, for both banks, if there are missionaries present on the bank they cannot be out numbered by cannibals (if they were the cannibals would eat the missionaries) The boat cannot cross the river by itself with no people on boat.

→ 1] state description :

(x_a, y_a) $(x_b y_b), Z)$

x_a = no. of M on bank A

y_a = no. of C on bank A

x_b = no. of M on bank B

y_b = no. of C on bank B

Z = boat is at which bank.

2] Step 2 :

Initial state : [(3,3), (0,0), A]

Final state : [(0,0), (3,3), B]

Step 3 : Actions

1 M - Send one missionary to other side of boat bank

1 C - send one cannibal to other side of boat bank

2 M - Send ~~one~~ 2 missionary to other side of boat bank

2 C - send 2 cannibals to other side of boat bank

1 M 1 C - send 1 missionary and 1 cannibal to other side of boat bank.

Step 4: Find soln

2C [(3,3), (0,0), A] ← IS

[(3,1), (0,2), B]

↓ 1C

[(3,2), (0,1), A]

↓ 2C

[(3,0), (0,3), B]

↓ 1C

[(3,1), (0,2), A]

↓ 2M

[(1,1), (2,2), B]

↓ 1M, 1C

[(2,2), (1,1), A]

↓ 2M

[(0,2), (3,1), B]

↓ 1C

[(0,3), (3,0), A]

↓ 2C

[(0,1), (3,2), B]

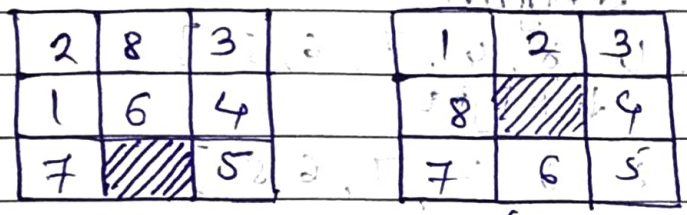
↓ 1C

[(0,2), (3,1), A]

↓ 2C

[(0,0), (3,3), B] ← GS

8 Puzzle Problem :



Step 1 : A state can be described as tile pos^n on the board and position of a blank. It can be represented as 9 integer vector.
The 1st integer in the vector represent the tile at 1st location & so on. blank can be represented as integer 0.

Step 2 : Initial state : [2, 8, 3, 1, 6, 4, 7, 0, 5]
Goal state : [1, 2, 3, 8, 0, 4, 7, 6, 5]

Step 3 : Actions
Assume 4 actions for movement of blank and not for tiles

pos^n of
Swap the blank and tile on the right
Swap the pos^n of blank and tile on the left
Swap the pos^n of blank and tile on the up
Swap the pos^n of blank and tile on the Dow

Step 4 : Sol^n

[2, 8, 3, 1, 6, 4, 7, 0, 5] ← Is

↓

[2, 8, 3, 1, 0, 4, 7, 6, 5]

↓ up

[2, 0, 3, 1, 8, 4, 7, 6, 5]

left

[0, 2, 3, 1, 8, 4, 7, 6, 5]

↓ Right

[1, 2, 3, 0, 8, 4, 7, 6, 5]

↓ Right

[1, 2, 3, 8, 0, 4, 7, 6, 5]

JMP

SM ① Describe Problem solving agent.

* Searching Techniques:

- Uninformed Searching (Blind)
- Informed Searching (Heuristic)

1) Uninformed Searching techniques:
- These techniques are strategy oriented While following the strategy, they do not have any additional information about the goal. Hence they are not goal oriented.

- These techniques, ~~for~~ follow the goal blindly. Therefore they are called as Blind search techniques.

2) Informed Searching Techniques:
- These techniques are goal oriented. They use the description of the goal while following the strategy.

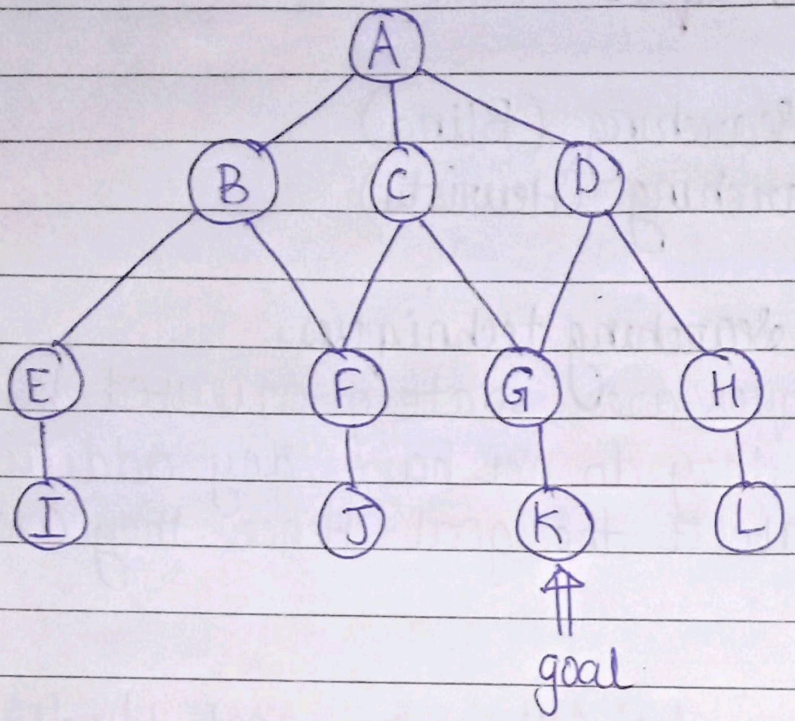
- Before moving ahead, they estimate the distance of goal state from each state and set the preference to visit the next state.

- Estimating the distance of goal state from any state is called as heuristic function (evaluation)

1) Blind
→ Types of uninformed Search techniques:

1) Breadth first search
2) Depth first search
3) Depth limited search
4) Iterative deepening search
5) Uniform cost search (UCS)

State space problems → can be expressed in the form of graph



1) BFS:

(also called as fringe leaves)

1) CLOSE = [ ] ; OPEN = [A]
2) ~~CLOSE = [ ]~~ X = A ; CLOSE = [A] ; OPEN = [B, C, D]
3) X = B ; CLOSE = [A, B] ; OPEN = [C, D, E, F]
4) X = C ; CLOSE = [A, B, C] ; OPEN = [D, E, F, G]
5) X = D ; CLOSE = [A, B, C, D] ; OPEN = [E, F, G, H]
6) X = E , CLOSE = [A, B, C, D, E] ; OPEN = [F, G, H, I]
7) X = F ; CLOSE = [A, B, C, D, E, F] ; OPEN = [G, H, I, J]
8) X = G ; CLOSE = [A, B, C, D, E, F, G] ; OPEN = [H, I, J, K]
9) X = H ; CLOSE = [A, B, C, D, E, F, G, H] ; OPEN = [I, J, K, L]
10) X = I ; CLOSE = [A, B, C, D, E, F, G, H, I] ; OPEN = [J, K, L]
11) X = J ; CLOSE = [A, B, C, D, E, F, G, H, I, J] ; OPEN = [K, L]
12) X = K ; CLOSE = [A, B, C, D, E, F, G, H, I, J, K] ; OPEN = [L]

goal

2) DFS:

1) CLOSE = [ ] ; OPEN = [A]
2) X = A ; CLOSE = [A] ; OPEN = [B]
3) X = B ; CLOSE = [A, B] ; OPEN = [E]
4) X = E ; CLOSE = [A, B, E] ; OPEN = [I]
5) X = I ; CLOSE = [A, B, E, I] ; OPEN = [F]

1) CLOSE = [ ] ; OPEN = [A]
2) X = A ; CLOSE = [A] ; OPEN = [B, C, D]
3) X = B ; CLOSE = [A, B] ; OPEN = [E, F, C, D]
4) X = E ; CLOSE = [A, B, E] ; OPEN = [I, F, C, D]
5) X = I ; CLOSE = [A, B, E, I] ; OPEN = [F, C, D]
6) X = F ; CLOSE = [A, B, E, I, F] ; OPEN = [J, C, D]
7) X = J ; CLOSE = [A, B, E, I, F, J] ; OPEN = [C, D]
8) X = C ; CLOSE = [A, B E, I, F, J, C] ; OPEN = [G, D]
9) X = G ; CLOSE = [A, B, E, I, F, J, C] ; OPEN = [K, D]
10) X = K ; CLOSE = [A, B, E, I, F, J, C, K] ; OPEN = [D]

→ Algorithm for BFS:

1] Create a single member queue comprising of root node.

2] If the first member of the queue is the goal node then go to step 5.

3] If the first member of the queue is not a goal node then remove it from the queue and add it to the list of visited nodes.
Consider its children nodes if any and add them to the rear end of the queue.

4] If the queue is empty, then go to step 6 else, go to step 2

5] Print "Success" and stop.

6] Print "failed" and stop.

→ Algorithm for DFS:

1] Create a single member queue comprising of root node.

2] If the first member of the queue is the goal node, then go to step 5.

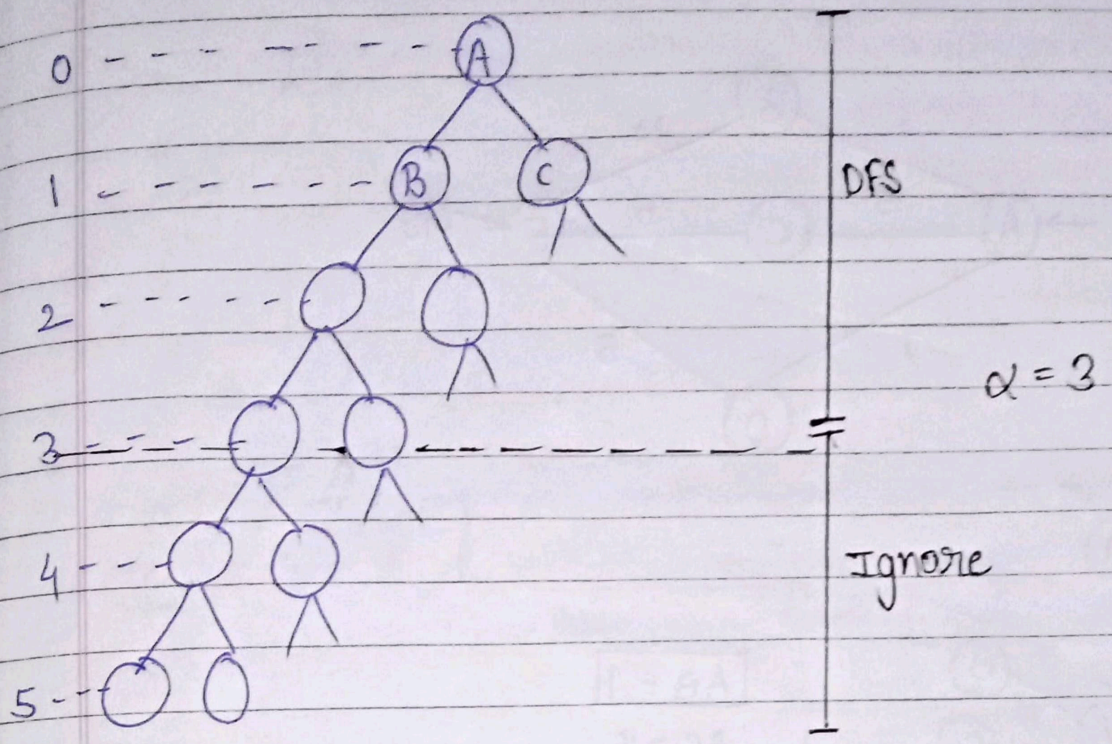
3] If the first member of the queue is not the goal node, then remove it from the queue and add it to the list of visited nodes.
Consider its children nodes if any and add them to the front end of the queue.

4] If the queue is empty, then go to step 6 else, go to step 2.

5] Print "Success" and stop.

6] Print "failed" and stop

3) Depth limited Search:

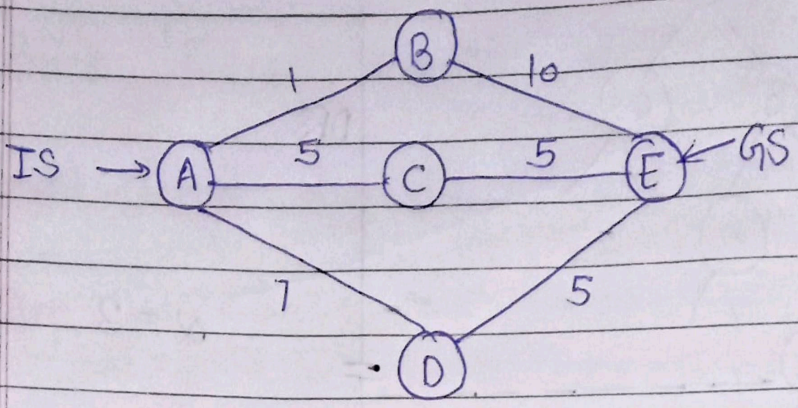


- It is a variation of depth first search. In this, we can set some limit to the depth (α)
- All the nodes at level α are assumed as ~~tet~~ leaf nodes and DFS is implemented only upto the nodes at level α, rest of the nodes are ignored

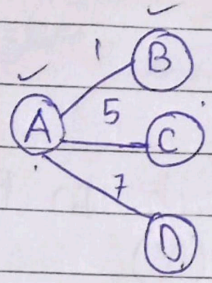
4) Iterative deepening DFS:

- It is the implementation of Depth limited search in iterations. For every iteration, we can set some limit to the depth i.e α.
- If the goal is not achieved, then we can increase α by some constant.
- The iterations will continue till either the goal is achieved or complete tree is searched.

5) Uniform cost search (UCS)



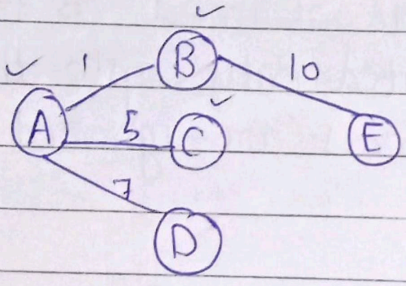
1) A

2) 

AB = 1

AC = 5

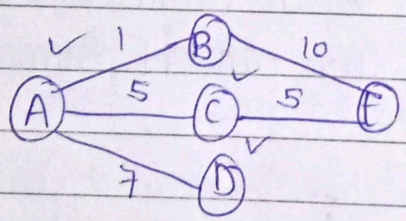
AD = 7

3) 

ABE = 11

AC = 5

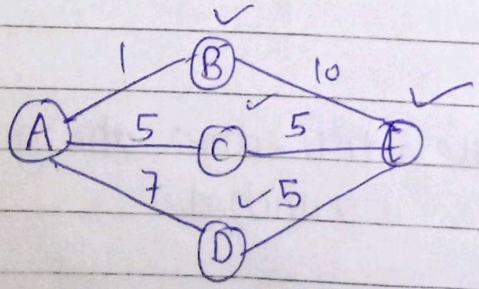
AD = 7

4) 

ABE = 11

ACE = 10

AD = 7

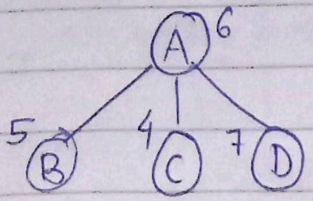
5) 

ABE = 11

ACE = 10

ADE = 12

2) Heuristic function :



$h(n) = k$

state — heuristic value

eg) $h(A) = 6$
$h(C) = 4$

It is used to make the searching more efficient

→ 8 puzzle problem :

2	8	3
1	6	4
7		5

IS

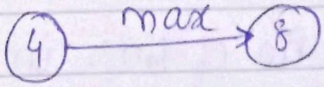
1	2	3
8		4
7	6	5

GS

- First heuristic function for 8 puzzle :

1) Count the number of tiles in their correct place in a state as per the goal state.

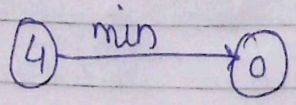
$h(IS) = 4$
$h(GS) = 8$



- Second heuristic function :
Count the number of tiles which are incorrectly placed

in a state as per the goal state...

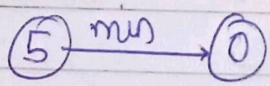
h(IS) = 4
h(GS) = 0



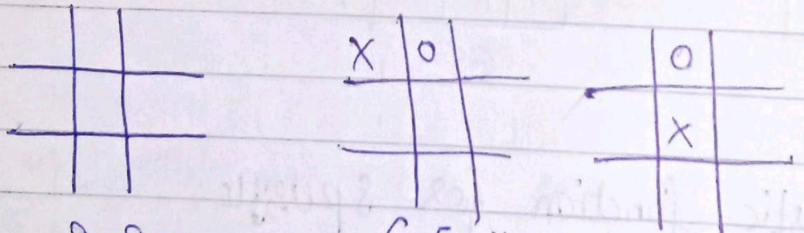
- Third heuristic distance:

Calculate the manhattan distance for a state in which, count how far each tile is from its correct place as per goal and add.

h(IS) = 5
h(GS) = 0



→ Tic Tac Toe



Heuristic function for tic tac toe is

h(n) = x-y

where x is no of lines in which you can win and y is no. of lines where opponent can win.

X	O	
X		

O	O	
	X	

"$h(n) = (3 * X_2 + X_1) - (3 * O_2 + O_1)$"

$\therefore h(n) = (3*1+2) - (3*0+1)$
$= 5 - 1$
$= 4$

$h(n) = (3*0+2) - [3*1+1]$
$= 2 - 4$
$= -2$

→ Blocks world problem:

-7	-1	A
-6	-1	H
-5	-1	G
-4	-1	F
-3	-1	E
-2	-1	D
-1	-1	C
0	-1	B

H	7
G	6
F	5
E	4
D	3
C	2
B	1
A	0

- First Heuristic function:

1) Add -1 point for every block that is resting on the correct thing as per the goal.

2) Subtract 1 point for every block that is resting on wrong thing as per the goal.

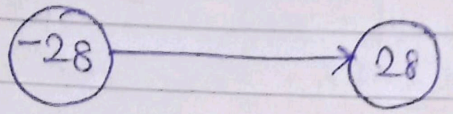
$h(S) = 4$
$h(GS) = 8$

(4) → (8)

- 2nd heuristic function.
  - for each block that has a correct block structure, add 1 point in the existing support structure.
  - for each block that has incorrect support structure, subtract 1 point for every support structure

$h(IS) = -28$

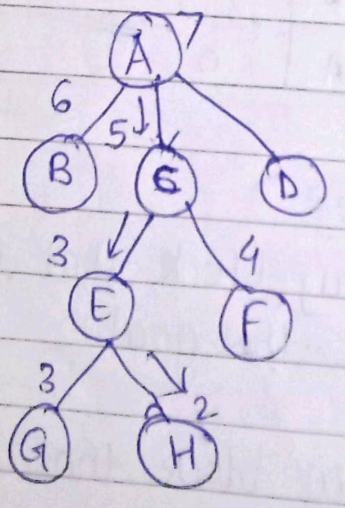
$h(GS) = 28$

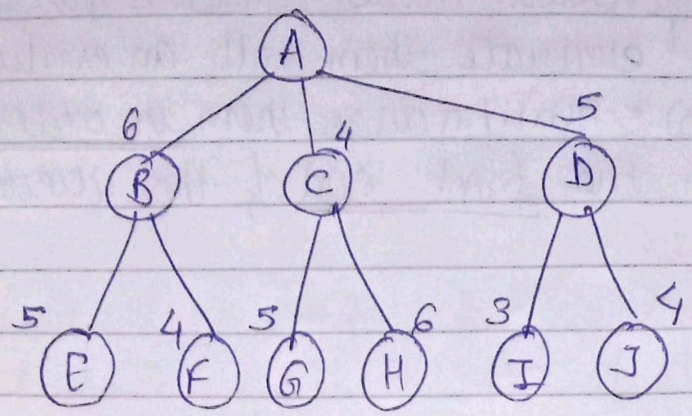


* Informed search techniques:

1) Hill climbing search
2) Best First search
3) A* search.

1) Hill climbing search.





A C G H D I J B F E

1) CLOSE = [ ] , OPEN = [A7]
2) ~~CLOSE =~~ X = A , CLOSE = [A] , OPEN = [C4, D5, B6]
3) X = C ; CLOSE = [A, C] ; OPEN = [G5, H6, D5, B6]
4) X = G ; CLOSE = [A, C, G] ; OPEN = [H6, D5, B6]
5) X = H ; CLOSE = [A, C, G, H] ; OPEN = [~~H6~~, D5, B6]
6) X = D ; CLOSE = [A, C, G, H, D] ; OPEN = [I3, J4, B6]
7) X = I ; CLOSE = [A, C, G, H, D, I] ; OPEN = [J4, B6]
8) X = J ; CLOSE = [A, C, G, H, D, I, J] ; OPEN = [B6]
9) X = B ; CLOSE = [A, C, G, H, D, I, J, B] ; OPEN = [F4, E5]
10) X = F ; CLOSE = [A, C, G, H, D, I, J, B, F] ; OPEN = [E5]
11) X = E ; CLOSE = [A, C, G, H, D, I, J, B, F] ; OPEN = [ ]

⇒ Algorithm for Hill Climbing:

Same as DFS

1]

2]

3] If the first member of the queue is not a goal node then remove it from queue and add it

to the list of visited nodes. Consider its children nodes if any, evaluate them with an evaluation function $f(n) = h(n)$. Arrange them in order and add them to the front end of the queue.

same as DFS

4]

5]

6]

⇒ Types of Hill climbing:

1) Steepest Ascent Hill climbing

1] Generate all the successors of current state

2) Select the best successor as a next state to visit

3] Go to step 1 until either the goal is achieved or the dead end is reached

4] If the goal is acheived then exit the search. In case of dead end, back track to an earlier state and select the next best successor as a next state to visit and go to step 1.

2) Simple Hill climbing:

1] Generate the first successor of the current state.

2] If it is better than the current state then make that successor as the current state. Otherwise, generate the next successor of the current state.

3] This will continue until the goal is achieved or the dead end is reached. In both the cases, the search terminates

⇒ Solve 8 puzzle using hill climbing.

2	8	3
1	6	4
7		5

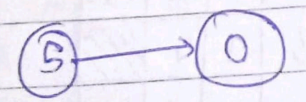
IS

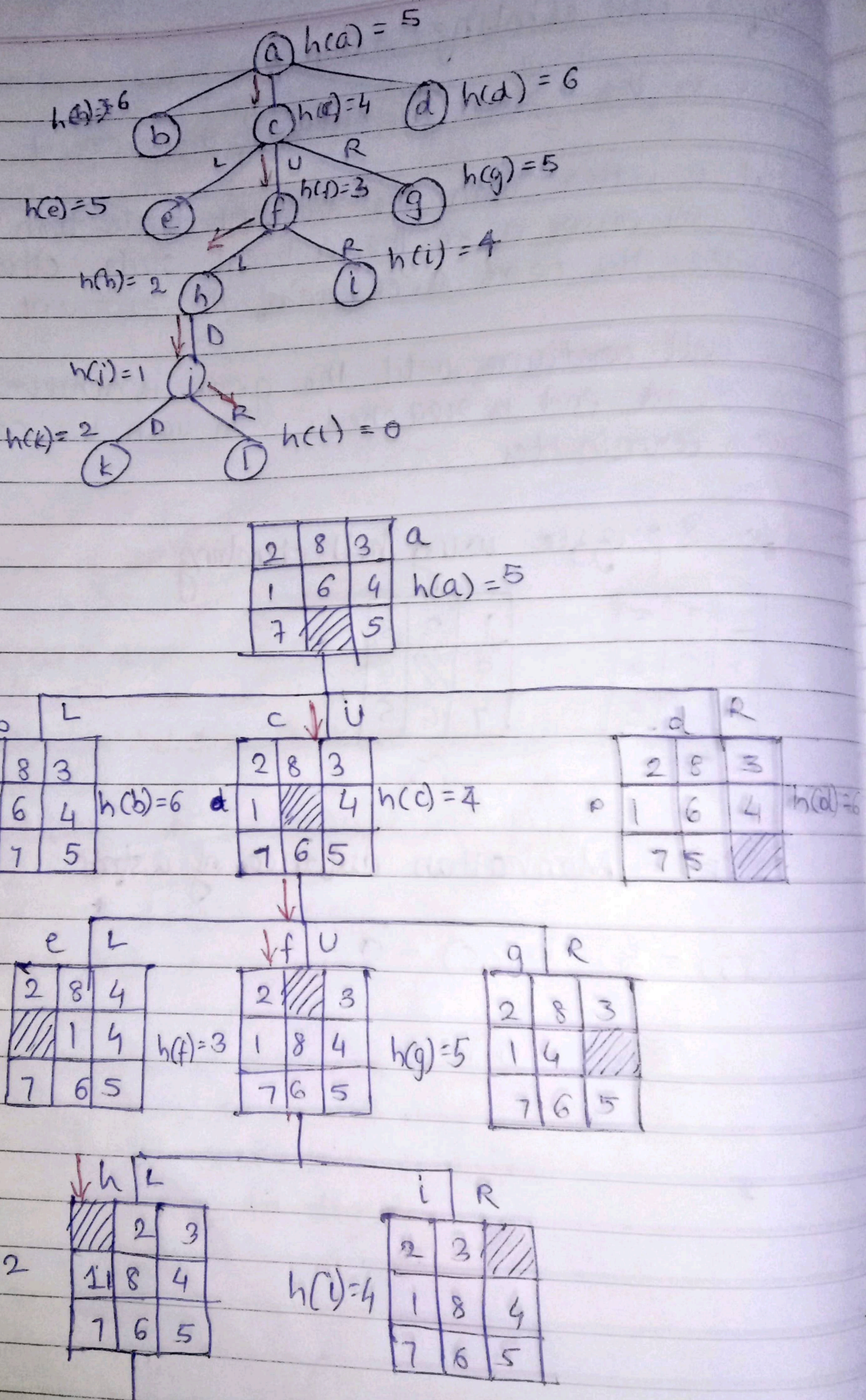
1	2	3
8		4
7	6	5

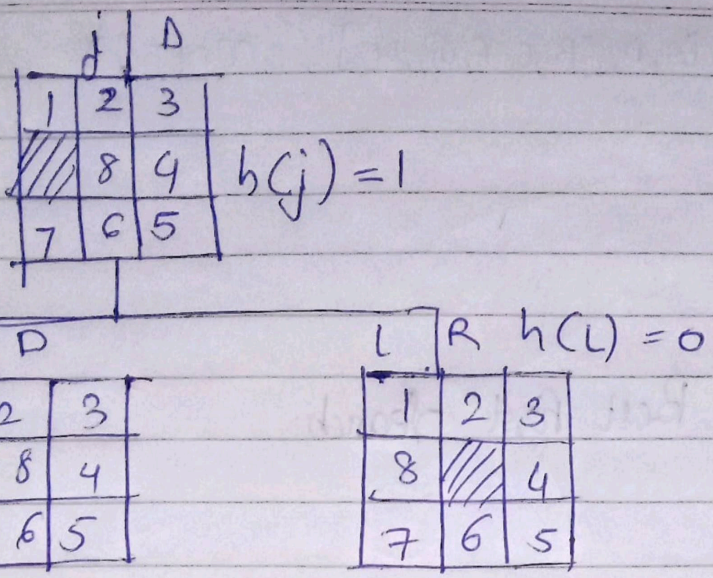
GS

h(n) = Manhattan distance of a state

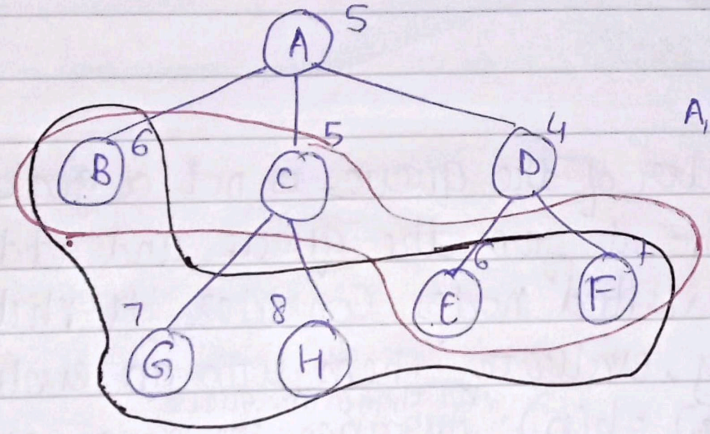
h(IS) = 5 h(GS) = 0




a h(a) = 5
b h(b) = 6
c h(c) = 4
d h(d) = 6
e h(e) = 5
f h(f) = 3
g h(g) = 5
h h(h) = 2
i h(i) = 4
j h(j) = 1
k h(k) = 2
l h(l) = 0
L
U
R
D
2 8 3
1 6 4
7 5
h(a) = 5
h(b) = 6
h(c) = 4
h(d) = 6
h(e) = 5
h(f) = 3
h(g) = 5
h(h) = 2
h(i) = 4



2) Best first Search :



A, D, C, B, E, F, G, H

It is an improvement over hill climbing. It will always select the next node to visit from list of all the nodes which are not yet visited i.e, the best of all the remaining nodes.

1. CLOSE = [ ] OPEN = [A5]
2. X = A ; CLOSE = [A] ; OPEN = [D4, C5, B6]
3. X = D ; CLOSE = [A, D] ; OPEN = [C5, B6, E6, F7]
4. X = C ; CLOSE = [A, D, C] ; OPEN = [B6, E6, ~~G7~~, F7, G7, H8]
5. X = B ; CLOSE = [A, D, C, B] ; OPEN = [E6, F7, G7, H8]
6. X = E ; CLOSE = [A, D, C, B, E], ; OPEN = [F7, G7, H8]
7. X = F ; CLOSE = [A, D, C, B, E] ; OPEN = [G7, H8]
8. X = G ; CLOSE = [A, D, C, B, E, ~~G~~ F, G] ; OPEN = [H8]

X=H ; CLOSE = [A, D, B, E, F, G, H] ; OPEN = [ ]

⇒ Algorithm for Best first search:

Same as DFS

1]

2]

3] If first member of the queue is not a goal node then remove ~~remo~~ it from the queue and add it to the list of visited nodes Consider its children nodes if any, evaluate them with an evaluation function $f(n) = h(n)$. Add them in the queue. Arrange them in order as the best will be the leftmost

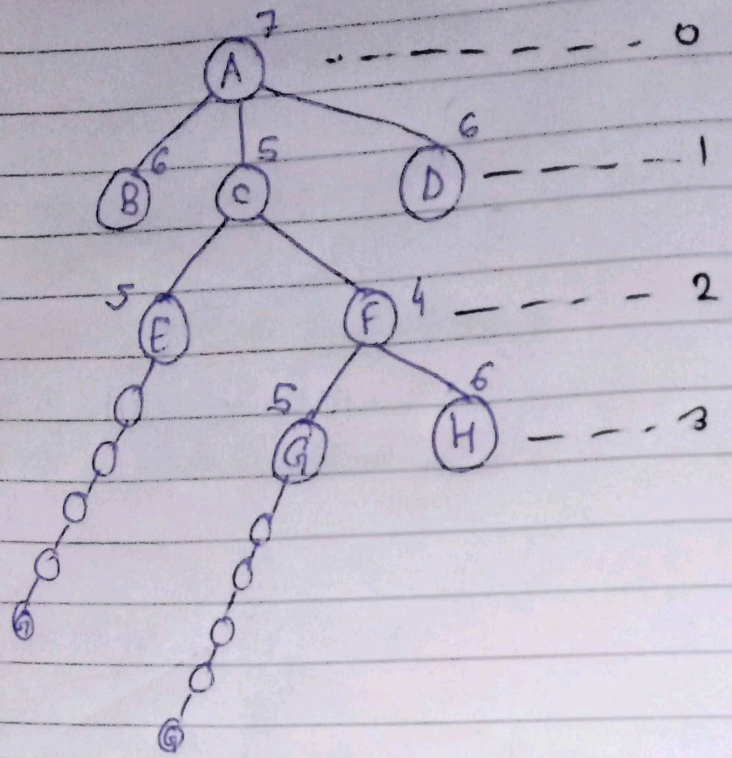
Same as DFS

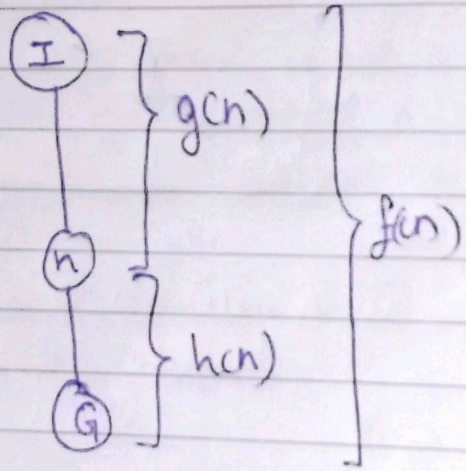
4)

5]

6)

* A* search:





Best first search : $f(n) = h(n)$

A^* : $f(n) = g(n) + h(n)$

- In Best First Search, if two states with same heuristic value appears ~~at~~ at two different level are not comparable.

- We can make them comparable by considering the distance of these states from the initial state along with the heuristic value.

Hence we get a new evaluation funct^n i.e

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

where, $g(n)$ is the distance of state n from initial state, $h(n)$ is the estimated distance of goal state from state n and $f(n)$ is the complete distance from initial state to goal state through state n.

By changing the evaluation function, we get new search i.e A* ~~search~~ search A* means Admissible Best first Search.

Admissibility is the property of an algorithm which guarantees optimal solution

→ Algorithm for A*.

1]

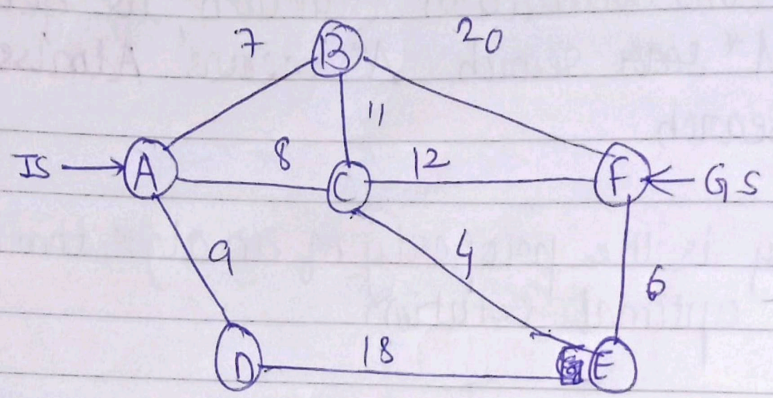
2]

3] If the first member of the queue is not a goal node then remove it from the queue and add it to the list of visited nodes. Consider its children nodes if any, evaluate them with an evaluation function $f(n) = g(n) + h(n)$. Add them in the queue. Rearrange them in order as the test will be the leftmost
↳ with heuristic merit (best leftmost).

4)

5)

⇒ Route Finding Problem:



Q Find the route from city A to city F using A* and Best first search.

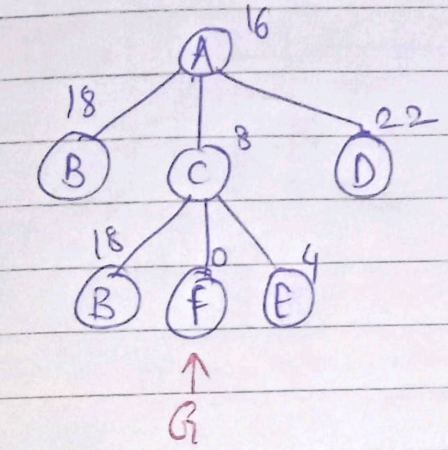
- For route finding problem, we consider the heuristic function as the straight line distance SLD from every city to city F. We assume that the straight line distance is less than actual distance by 2.

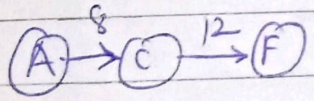
Following are the straight line distances from every city to city F.

A = 16
B = 18
C = 8
D = 22
E = 4
F = 0

A = ~~18~~ 16
B = 19
C = 10

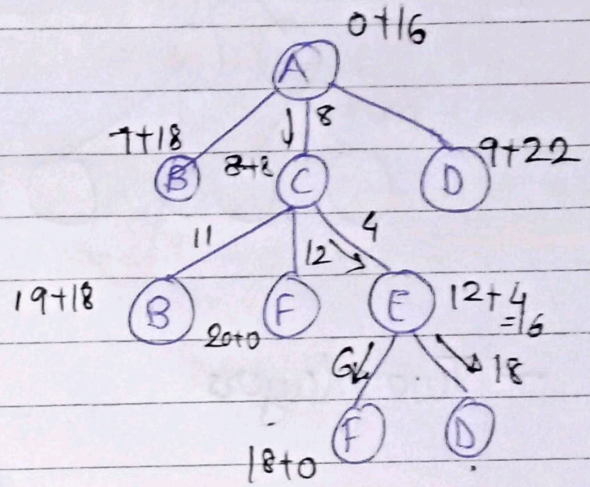
Best First Search

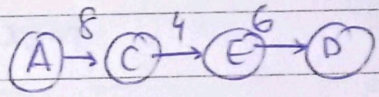




= 20

A*

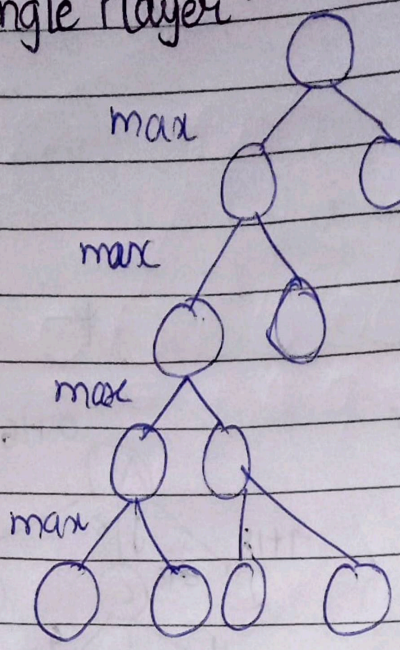




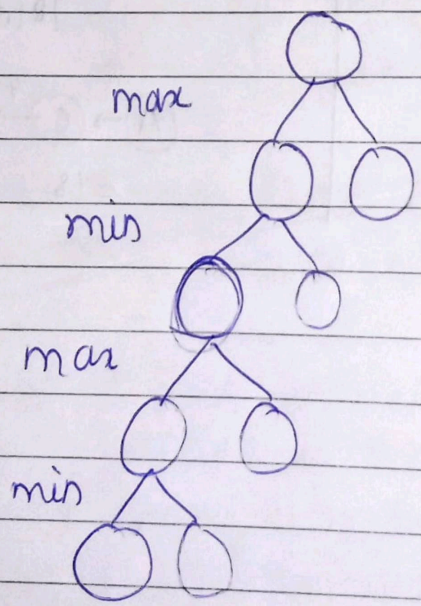
= 18

* Game Playing:

→ Single Player:

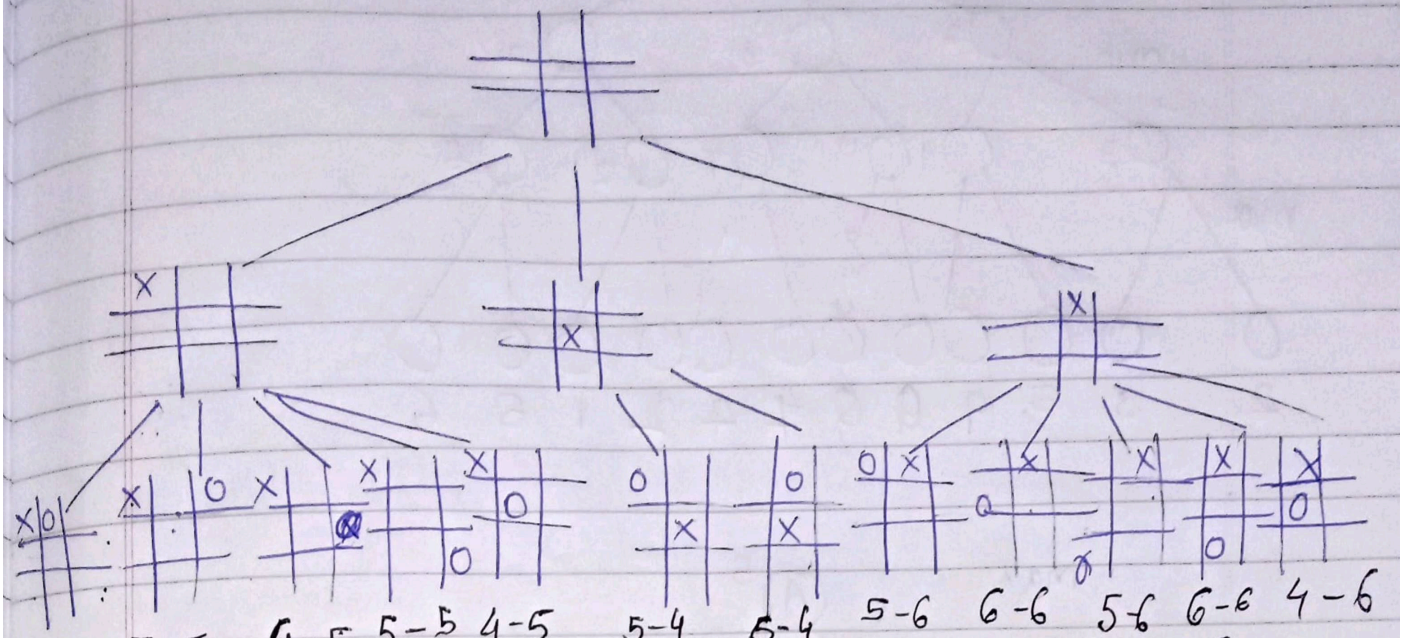


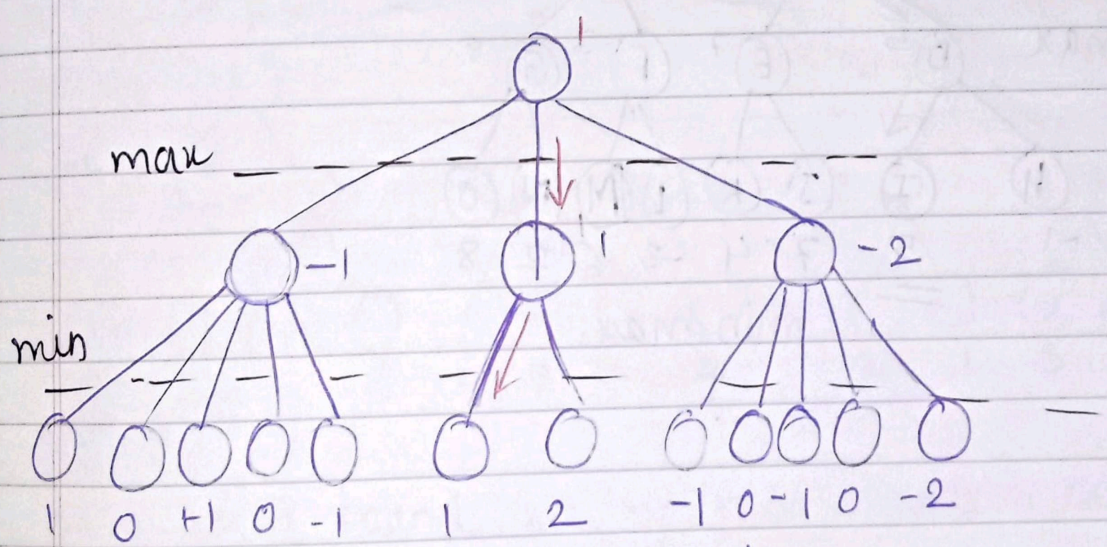
→ Two players:



- game tree [Min - max Search]

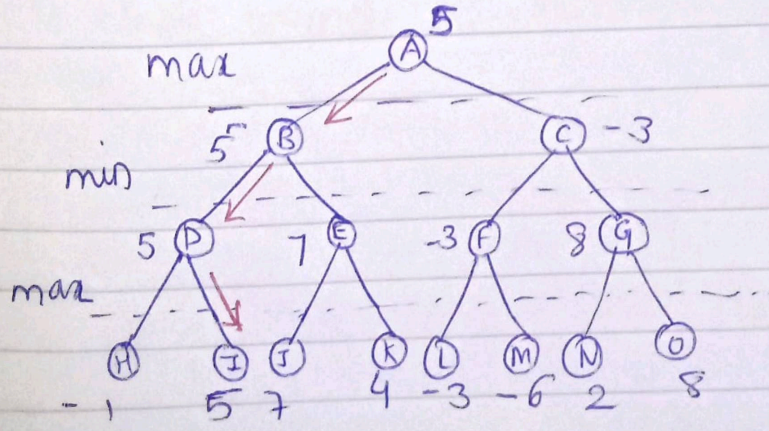
Q. Construct a game tree for tic-tac-toe:

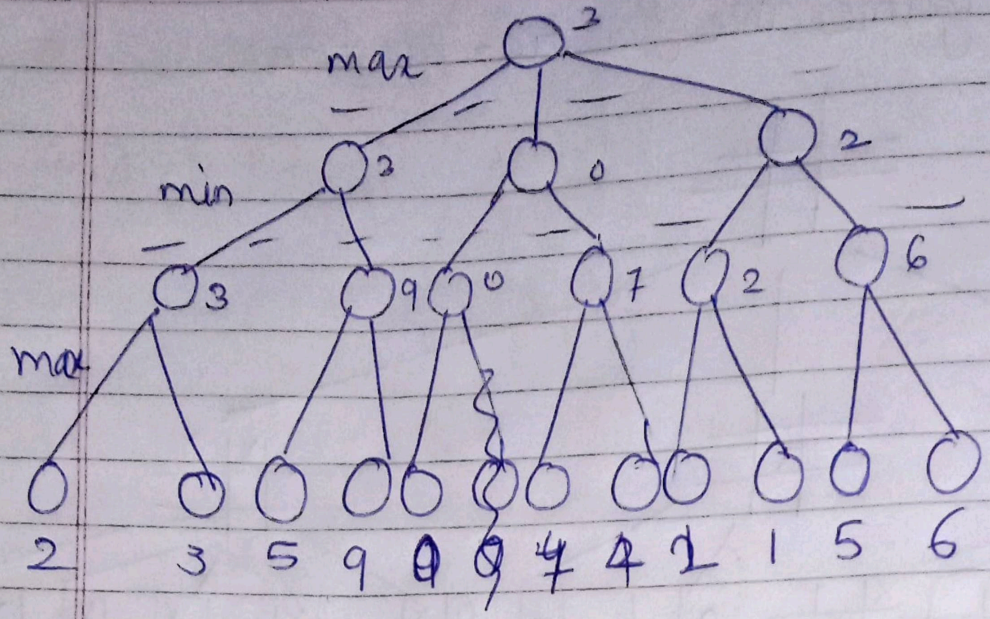


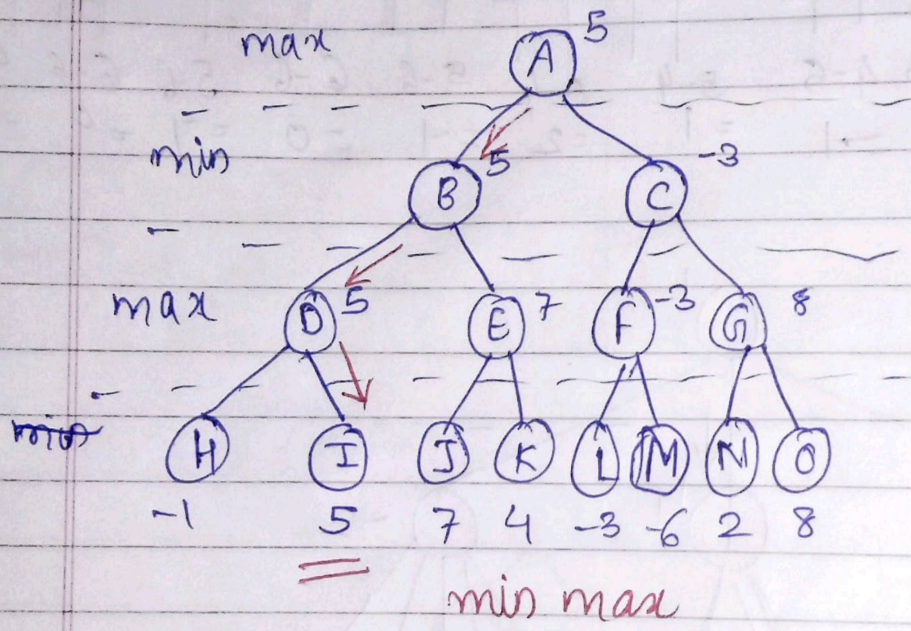


Min-max search

eg2)



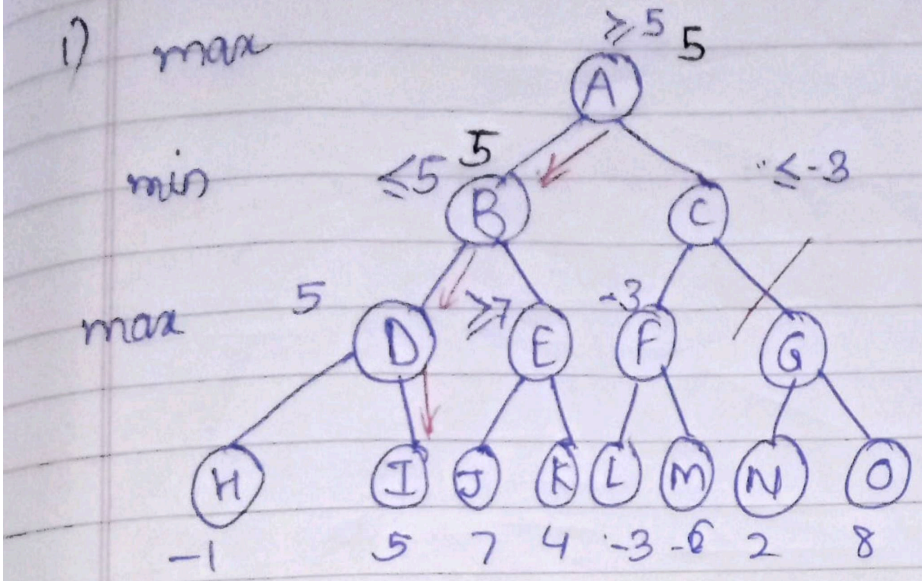

max
min
max
3
2
0
2
3
9
0
7
2
6
2 3 5 9 0 0 4 4 1 1 5 6


max
min
max
min
A 5
B 5
C -3
D 5
E 7
F -3
G 8
H
I
J
K
L
M
N
O
-1 5 7 4 -3 -6 2 8

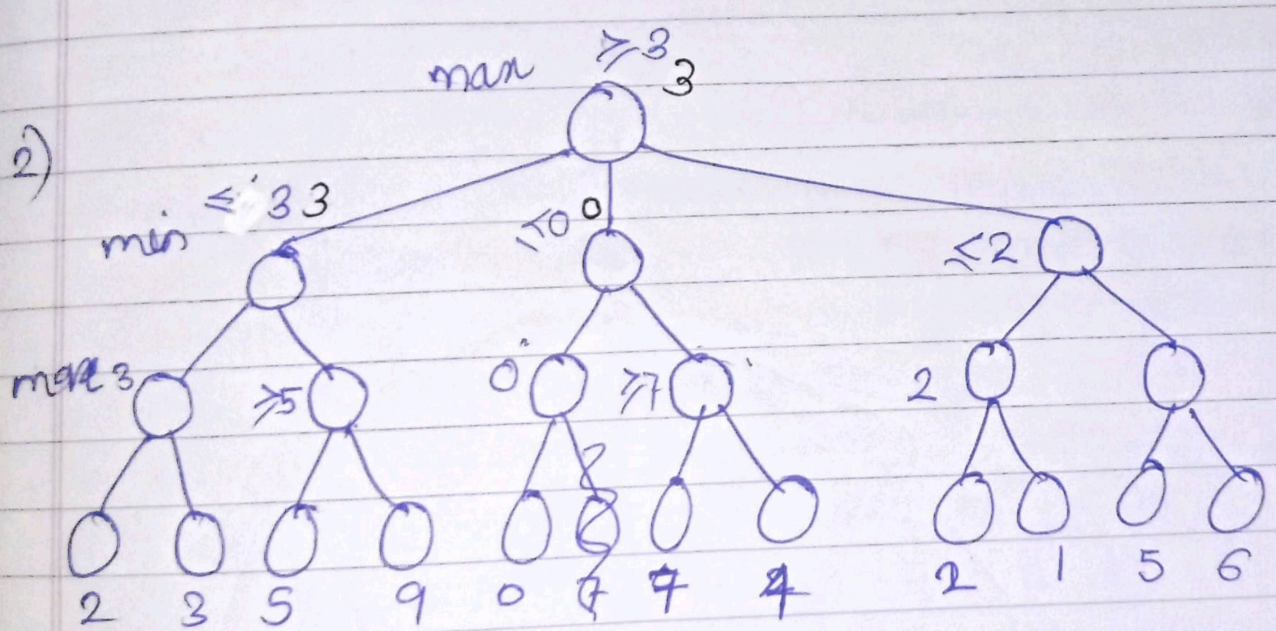
min max

* α-β pruning / α-β cutoff

1)

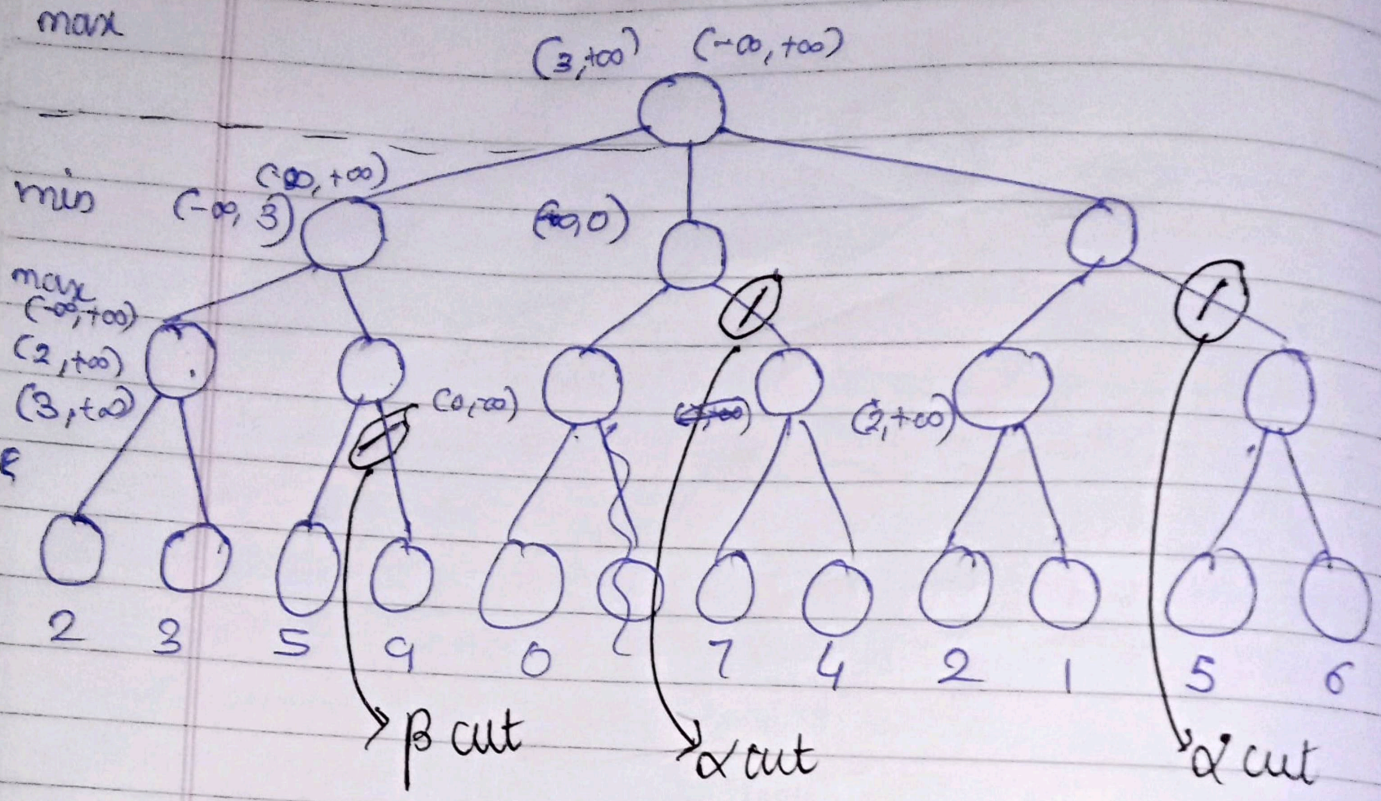


2)

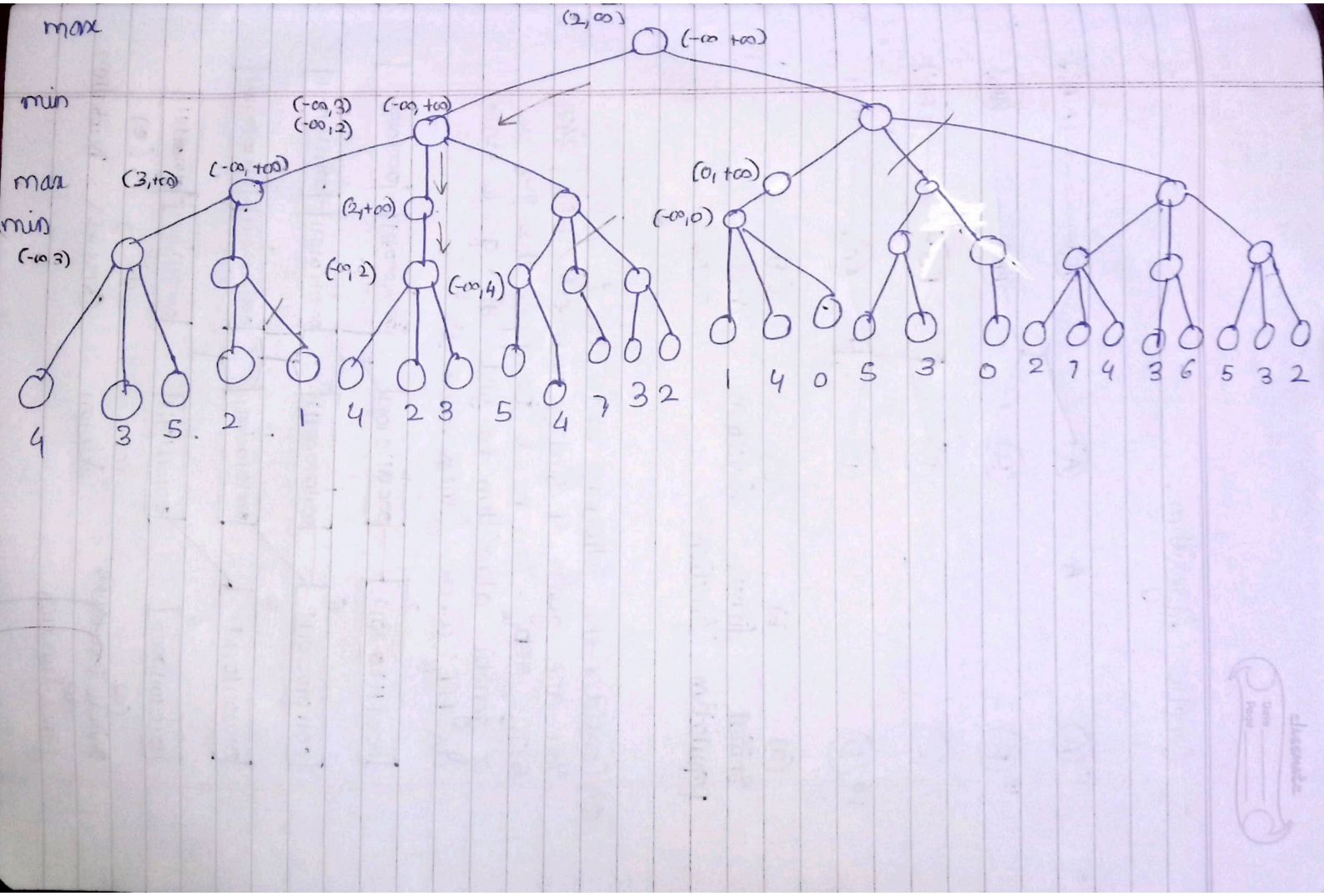


α → lower bound set by maximization node

β → Upper bound set by minimization node.



- Compare with grandparent node if condition exist
- ~~If~~ Visit all children of the leftmost node


max
min
max
min
(2, ∞)
(-∞ +∞)
(-∞, 3)
(-∞, 2)
(-∞, +∞)
(3, +∞)
(-∞, +∞)
(-∞ 3)
(2, +∞)
(-∞, 2)
(-∞, 4)
(0, +∞)
(-∞, 0)
4 3 5 2 1 4 2 3 5 4 7 3 2 1 4 0 5 3 0 2 7 4 3 6 5 3 2

Roll No: 21102A0037
CMPN A TE

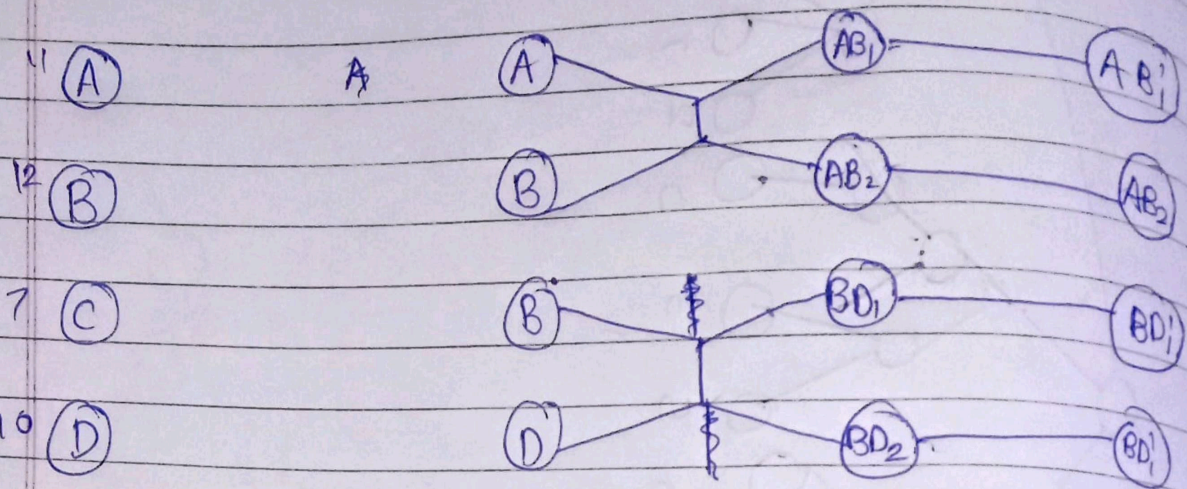
PAGE NO.: 7 | 2 | 24

1 Minute Paper

Q! write down difference between α & β cutoff and minmax search

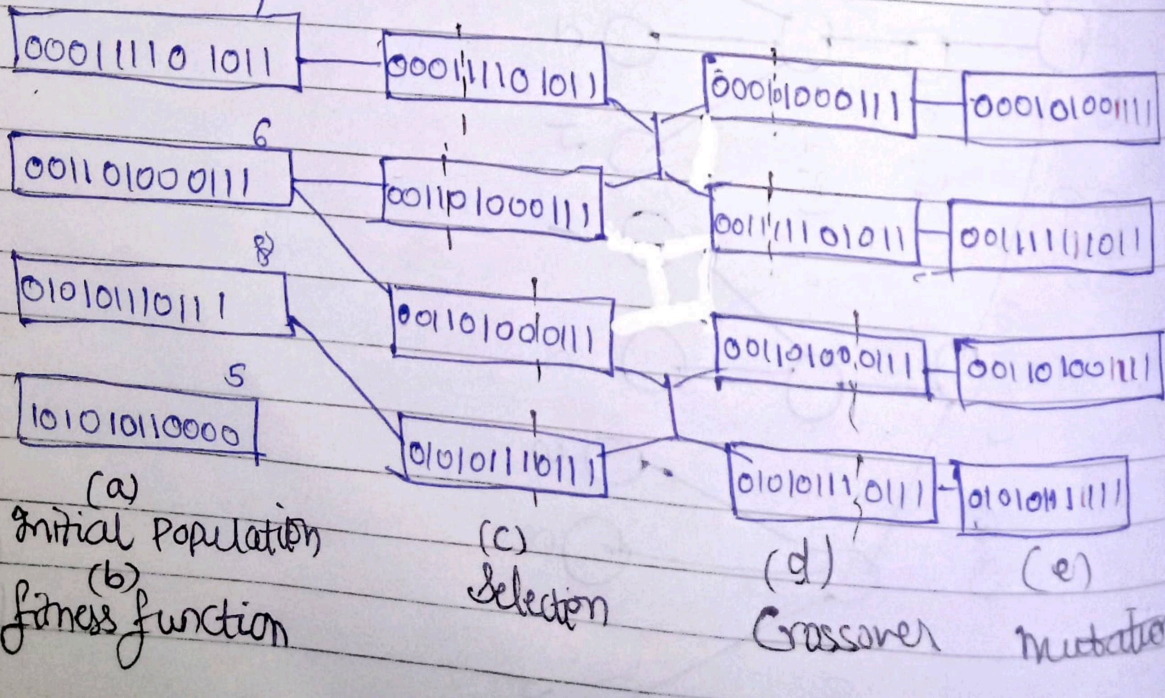
	MinMax search	α β cutoff
Defination	Algorithm for decision - making in games	Optimization techniques for minimax
objective	To find the best move for a player	To improve the efficiency of minmax
Evaluation	considers all possible moves and their outcomes	Reduces the number of evaluated moves
Pruning	Does not involve pruning	Involves pruning to eliminate branches
Complexity/ space	More complex and Required more memory to store entire game tree	less complex and Requires less memory due to pruning
Performance	slower due to exploring all possibilities	faster as it involve avoids unnecessary searches
Use cases	Suitable for small game trees / when optimal moves are essential	suitable for large game trees to improve search efficiency.

* Genetic Algorithm.

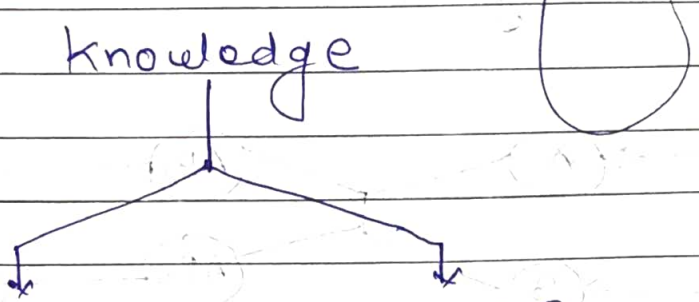


(a) Initial population (b) fitness function (c) Selection (d) Cross-overs

Q1) Consider the following example:-
You are given 4 binary numbers $1E.B_{16}$, 347_{16}, 527_{16}, ABD_{16}. You need to apply the process of genetic algorithm to find the goal state as FFF_{16} whose fitness value is 12.



4. Knowledge Representation

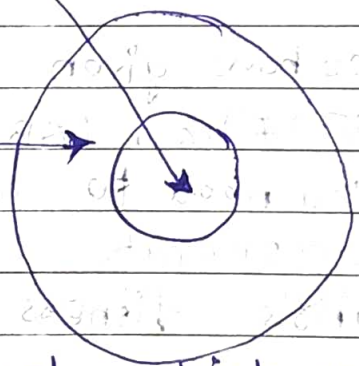


P P ⇐ declaration

Q P → Q Procedural knowledge

Types :

1) Propositional Logic :
2) Predicate logic



1) Propositional Logic

Proposition is a statement which can be either true/false but not both with respect to some content

Propositions are normally represented using symbol preferably "alphabet"

eg It is sunny - P (SUNNY)
It is raining - Q (RAINING)

Two or more proposition can be combined using logical connectives such as implication, double implication ($\leftrightarrow, \wedge, \vee, \neg$)
eg. If it is sunny then it is not raining
$P \rightarrow \neg Q$

consider the following facts and write them in propositional logic.
P - Rain humid - P HOT - Q
It is humid : P
If it is humid then it is Hot. $P \rightarrow Q$
If it is hot & humid then it will rain $P \wedge Q \rightarrow R$

Propositional is also called as Boolean LOGIC

2] Predicate Logic
(FOPL) First order Predicate logic
It is a language of logic it is more expresive way representing logical statement. ex chalk is white can be written as white(chalk)

Predicate logic uses one argument predicate 2 argument predicate, 3 argument predicate

1] single argument used to represent some extra info about object
eg Abhay is a boy. boy(abhay)

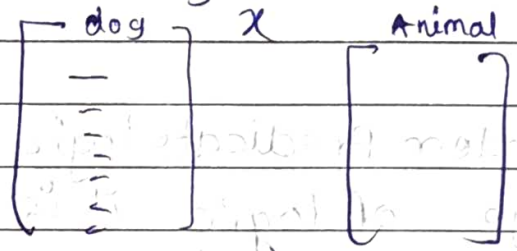
2) Two argument used to represent relation between 2 objects.
eg Abhay is a friend of Ajay friend(Abhay, Ajay)

3] Three Argument Predicate is used to represent some act
eg. Abhay gave book to Ajay.
Gave (abhay book Ajay)

convert the following statement into predicate logic

1) spot is a dog ⇒ Dog (spot)
2) Dog has a tail ⇒ Has a tail (Dog)
3) spot is tall ⇒ tall (spot)
4) All dogs are animal ⇒

$\forall x : [Dog(x) \rightarrow Animal(x)]$



5] All dogs have a tail Has a
$\forall x : [Dog(x) \rightarrow tail(x)]$
above eg is a example of Quantified Statement
Generalise statement about multiple object
the quantified statements can be represented in predicate logic only by using quantifier operators following 2 quantifiers are used in FOPL

1] Universal Quantifier ($\forall x$)
It is used to assert that the statement is true for all the values of associated variable.

eg All students are intelligent
$\forall x : [Student(x) \rightarrow Intelligent(x)]$

2] Existential Quantifier :
It is used to assert that there exist ∃ at least 1 Assignment for associated variable that will make Statement True.

eg Some students are intelligent
$\exists x : [student(x) \wedge Intelligent(x)]$

eg There is a Student
$\exists x : student(x)$

Some student are intelligent & smart
$\exists x : [student(x) \wedge intelligent \wedge smart]$

All Student are intelligent & smart
$\forall x : [Student(x) \rightarrow Intelligent \wedge smart]$

Nesting of all Quantifier

Everyone is loyal to someone
$\forall x : \exists y : Loyal(x,y)$
~~$\forall x : [$ everyone~~

Some is loyal to everyone
~~$\exists x : \forall x$~~ : Loyal(x,y)

everyone is loyal to everyone
~~$\forall(x,y) : loyal(x,y)$~~ $\forall x : \forall y : Loyal(x,y)$

Someone is loyal to someone $\exists x . \exists y \; loyal(x,y)$

The father of a son is also son of a father.

→

$$\forall x \left[\exists y \text{ father}(x, y) \rightarrow \left[\exists z : \text{son}(x, z) \right] \right]$$

$\forall x$

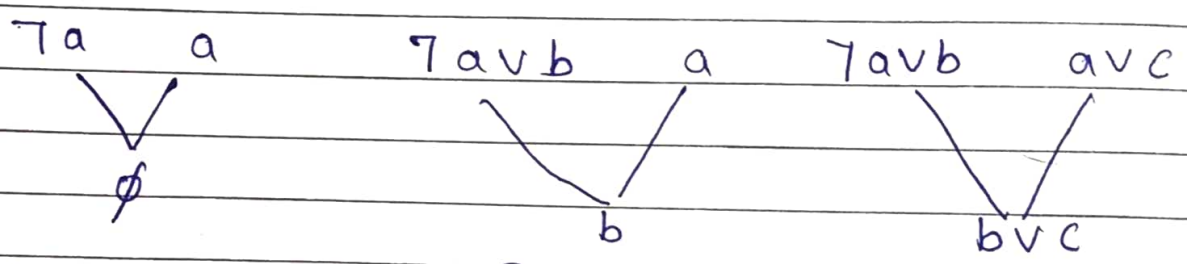
$\exists y$ father(x, y)

$\exists z$ son(x, z)

Resolution :-

Resolution uses contradiction it is a mechanism to draw some conclusion from Given set of facts it is also called as proof by refutation which means action of proving a statement or theory to be wrong / false.

Basic Resolution Rules :



Steps in Resolution Process :-

1) convert the given statement to either proportinal or predicate logic.
2) convert the logical statement to CNF (conjective normal form) / clause form.
3) Negate the conclusion or the fact which is to be resolved.
4) resolve the negation of conclusion is not true using refutation.

Conjective Normal form :-

logical statement may consisted of $[\forall, \exists, \rightarrow, \wedge, \vee, \neg]$ by eliminating $\forall, \exists, \rightarrow, \wedge$ the statement is converted to CNF.

PAGE NO.:

eliminate '→'

$a \rightarrow b = \neg a \vee b$

$a \wedge b \rightarrow c = \neg(a \wedge b) \vee c$

$= \neg a \vee \neg b \vee c$

eliminate '∧'

$a \wedge b = a$, b

$(a \vee b) \wedge c = a \vee b$, c

eliminate ∃ [skolemization]

1] $\exists x : P(x) = P(S)$ — Skolem function / Skolem constant

2] $\forall x : \exists y : P(x, y) = \forall x : P(x, S(x))$ — (skolem function)

eg someone is teacher

$\exists x : [\ teacher(x)\ ]$

|

Teacher (S)

eg Everyone is loyal to someone

$\forall x : \exists y : Loyal(x, y)$

$\forall x . Loyar(x, S(x))$

someone is loyal to everyone

$\exists x : \forall y : loyal(x, y)$

$\exists x : loyal\ to\ (S(x), y)$

Someone is loyal to someone

$\exists x : \exists y \; loyalto(x, y)$

$\exists y : loyalto(S_1, S_2(S_1))$

Eliminate $\forall$

$\forall x : P(x)$

$P(x)$

$\forall x : \forall y : P(x, y)$

Prefix → $P(x, y)$

normal form

e.g All are students

$\forall x : Student(x)$

↓

$Student(x)$

e.g every teacher is friendly with every student

$\forall x : \forall y : Teacher(x) \wedge Student(y) \rightarrow friendly(x, y)$

$(a \wedge b) \rightarrow c$

$\forall x : \forall y : \neg Teacher(x) \vee \neg Student(y) \vee friendly(x, y)$

$\neg Teacher(x) \vee \neg Student(y) \; Friendly(x, y)$

Q. consider the following facts
Rani is hungry.
If Rani is hungry she barks.
If Rani barking then raja gets angry.
Prove that

→ consider following propositional symbols

step 1 : P: Rani is hungry
Q: Rani is barking
R: Raja is angry

equivalent propositional logic statements

a: P
b: P → q
c: q → R

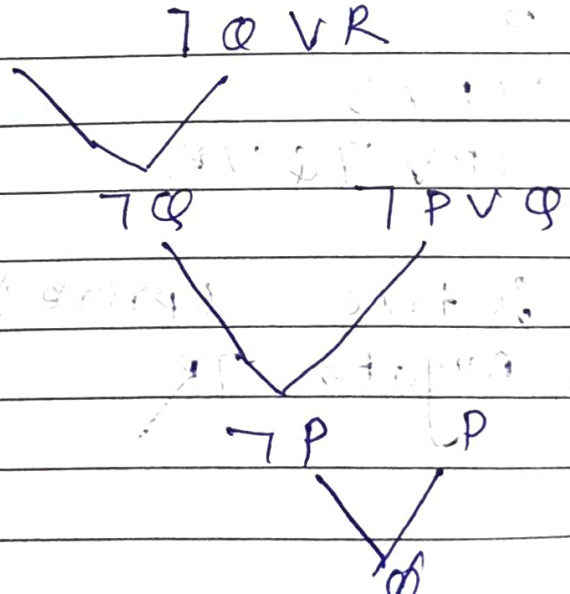
step 2: - convert the above in CNF

a: P
b: ¬P ∨ Q
c: ¬Q ∨ R

step 3: Prove R is true =>
Negate R : ¬R

step 4:

1)



We see that negation of conclusion that is ¬R has proved complete contradiction with given set of clauses (CNF) which means negation is completely invalid. Hence assertion is valid. That is R is true

Q consider following facts
It is humid.
If it is humid then it is hot
If it is hot and humid then it will rain.
Prove that It is raining.

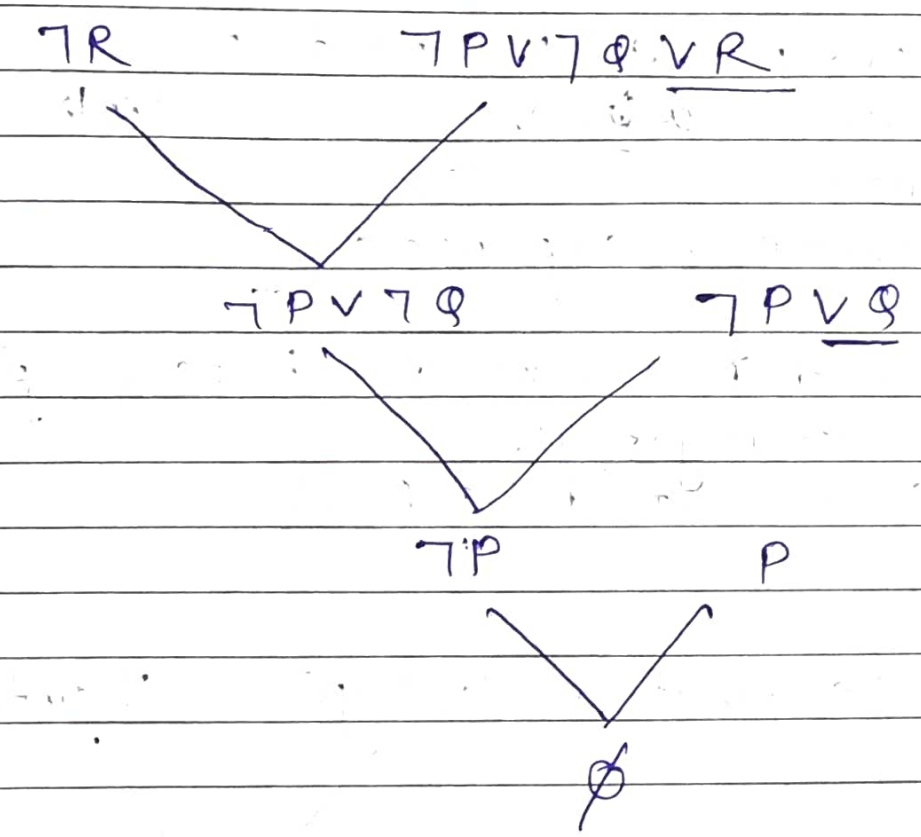
→

Consider the following propositional Symbol
Step 1: P: It is humid
Q: It is hot
R: It is raining

equivalent prop. logic statements
a: P
b: P → Q
c: P∧Q → R

Step 2: convert the above into CNF
a: P
b: ¬P ∨ Q
c: ¬P ∨ ¬Q ∨ R

Step 3: R is true (prove)
negate ¬R



Not for MSE

∴ Assertion is valid that is R is true

Q.3 Consider following facts

1) If A is on top of B then B support A
2) If A is above B and they are touching each other then A is on top of B.
3) A cup is above the book
4) A cup is touching the book

∴ prove that book support the cup

→

i) ∀a on top (a,b)
∀b support (b,a)

ontop (a, b) → support (b, a)

1) ∀a : ∀b : ontop (a, b) → support (b, a)
2) ∀x : ∀y : Above (x,y) ∧ Touching (x,y) → ontop (x,y)
3) Above (cup, book)
4) Touching (cup, book)

CNF

Step 2) ¬ ontop (a, b) V support (b, a)

¬ Above (x, y) V ¬ Touching (x, y) V ontop (x, y)

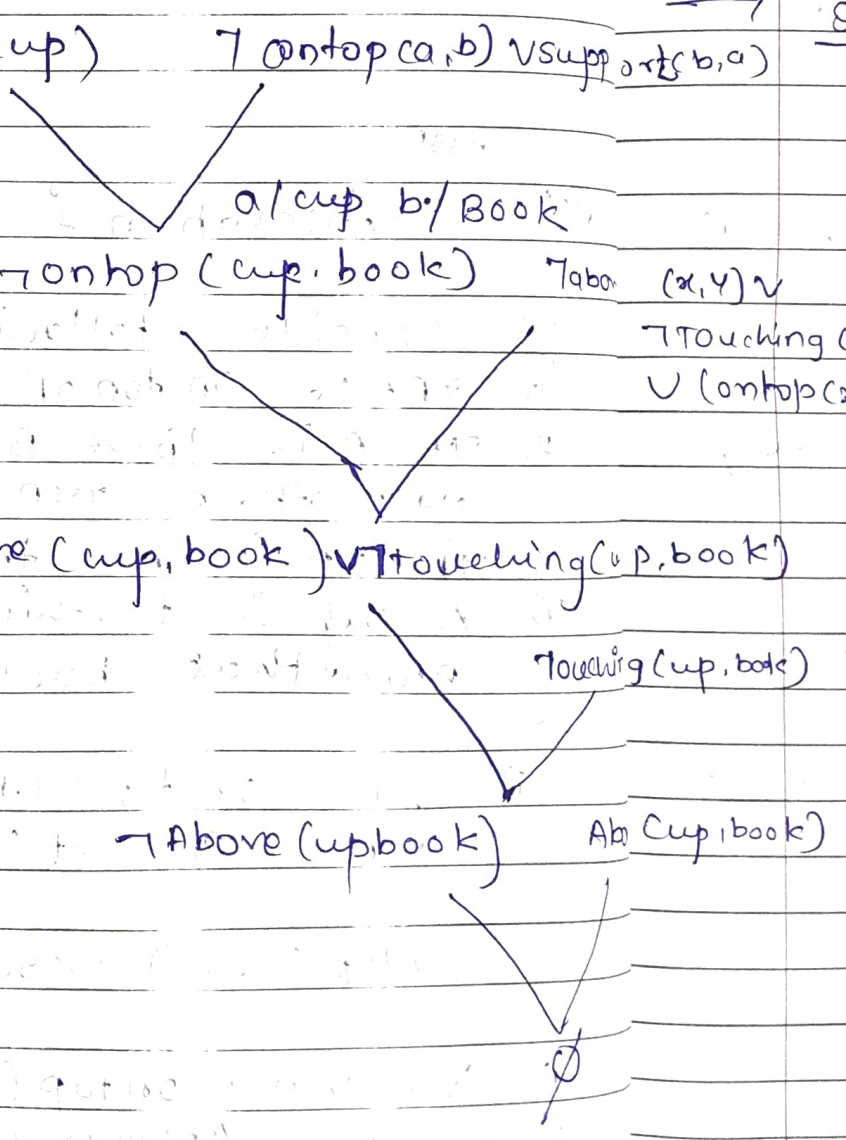
Above (cup, book)

Touching (cup, book)

Step 3: Prove that support (book (cup) is True

Take negation.

¬ support (book, cup)



Consider the following facts

1) Ravi likes all kind foods
2) Apple and chicken is food
3) Anything anyone eats and it not killed by is a food.
4) Ajay eats peanuts & is alive
5) Rita eats everything that Ajay eats

Prove that Ravi likes peanuts

→ Step 1:

a) $\forall x : food(x) \rightarrow likes(Ravi, x)$

$food(Apple) \wedge food(chicken)$

$\forall a : \forall b : Eats(a, b) \wedge \neg killed(a) \rightarrow food(b)$

$Eats(Ajay, Peanuts) \wedge Alive(Ajay)$

c) $\forall c : Eats(Ajay, c) \rightarrow Eats(Rita, c)$

6] Alive means not killed :

$\forall d : Alive(d) \rightarrow \neg killed(d)$

k) CNF :

Step 2 : $\neg food(x) \vee likes(Ravi, x)$

$food(Apple)$

$food(chicken)$

$\neg Eats(a, b) \vee killed(a) \vee food(b)$

$Eats(Ajay, Peanuts)$

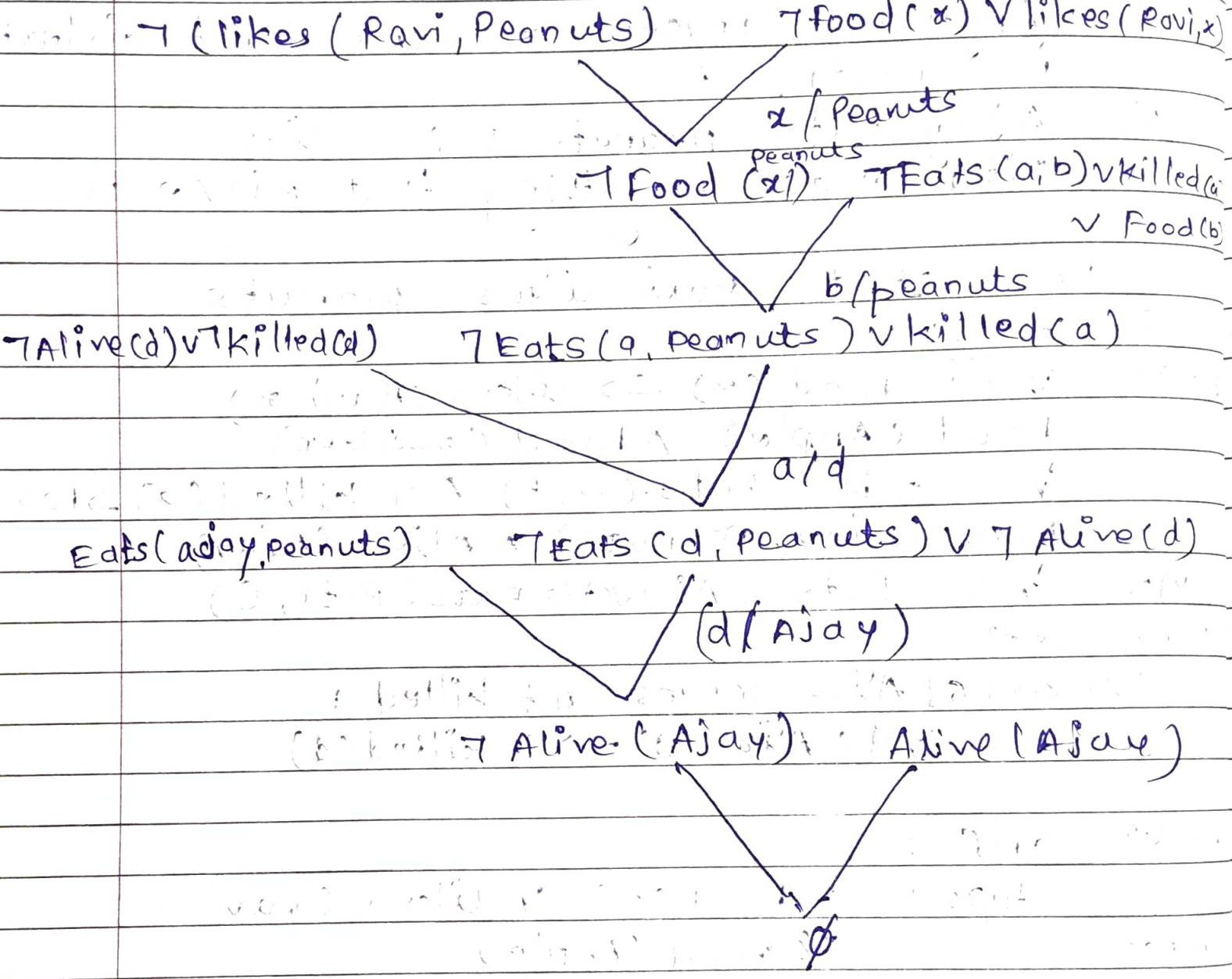
$Alive(Ajay)$

$\neg Eats(Ajay, c) \vee Eats(Rita, c)$

$\neg Alive(d) \vee \neg killed(d)$

To prove Ravi like peanuts is True

Negate ¬ likes (Ravi, peanuts)



Q. what food does rita eat. (Assume eats (Rita, z)

Steps to ~~Reduce~~ convert logical statement to CNF

1] Eliminate '$\rightarrow$'

$a \rightarrow b = \neg a \vee b$

2] Reduce the scope of Negation '$\neg$'

$\neg \neg P(x) = P(x)$

$\neg (a \wedge b) = \neg a \vee \neg b$

$\neg (a \vee b) = \neg a \wedge \neg b$

$\neg \forall x : P(x) = \exists x . \neg P(x)$

$\neg \exists x : P(x) = \forall x : \neg P(x)$

3] eliminate '$\exists$'

$\exists x : P(x) = P(s)$

$\forall x : \exists y : P(x, y)$

$\downarrow$

$\forall x : P(x, S(x))$

4] eliminate '$\forall$'

$\forall x : P(x) = P(x)$

$\forall x : \forall y : P(x, y) = P(x, y)$

5] eliminate $\wedge$

$P(x) \wedge Q(y) = P(x)\ Q(y)$

consider the following statement & convert it to CNF.

Everyone who loves animal is loved by someone

→ 1) eliminate →

$\forall x : [\forall y : Animal(y) \rightarrow loves(x,y)] \rightarrow [\exists z : loves(z,x)]$

$\forall x : \neg [\forall y : \neg Animal(y) \vee Loves(x,y)] \vee \exists z : loves(z,x)$

2) Reduce the scope of '¬'

$\forall x : \exists y : \neg(\neg Animal(y) \vee Loves(x,y)) \vee \exists z : loves(z,x)$

$\forall x : \exists y : (Animal(y) \wedge \neg loves(x,y)) \vee \exists z : loves(z,x)$

3) eliminate '∃'

$\forall x : (Animal(S(x)) \wedge \neg Loves(x, S(x))) \vee loves(S_2(x), x)$

4) eliminate '∀'

$(Animal(S_1(x)) \wedge \neg loves(S_1, S(x))) \vee loves(S_2(x), x)$

5] eliminate ∧

~~$Animal(S_1(x)) \wedge (\neg loves(S_1, S(x)) \vee loves(S_2(x), x))$~~

$Animal(S_1(x)) \vee loves(S_2(x), x)$

$\neg loves(x, S(x)) \vee loves(S_2(x), x)$

Uncertain knowledge representation

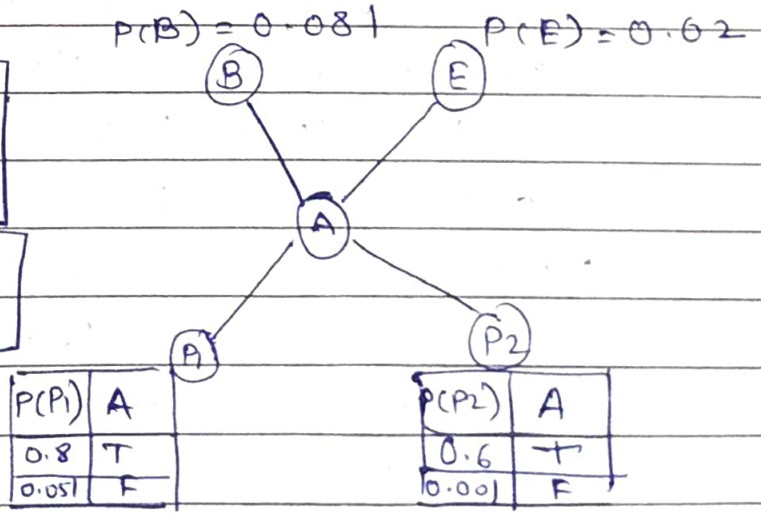
Joint probability distribution

$$P(x_1, x_2, x_3, \ldots, x_n) = \prod_{i=1}^{n} (P(x_i \mid \text{Parent}(x_i)))$$

Examples of uncertain domain
You have install burgler alarm at home which is fairly reliable to detect burglurery or minor earthquake. You have 2 neighbours P_1 & P_2 who promised you to call you at work when they hears the alarm. P_1 sometime forget to call you even he hears the alarm P_2 also sometimes will not call you because he confuse the alarm from specific distance.
what is a prob. that P_1 & P_2 calls you when they hears the alarm (alarm ringing) but no burglary & no earthquake.

→

burgalary (B)
Earthquake (E)
Alarm (A)
P_1 calls (P_1)
P_2 calls (P_2)



P(P1)	A
0.8	T
0.05	F

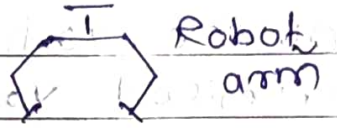
P(P2)	A
0.6	T
0.001	F

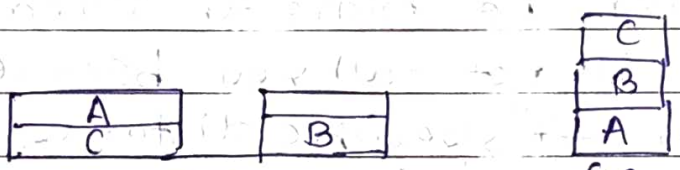
5. Planning

Plan consist of sequence of action to be taken to achieve the goal.
designing a planning agent needs the complete idea of any problem such as

1) identify the object and their way of representation.
2) identify the operators (actions) with their preconditions and effect

example: provide aplan for the blocks world problem





Step 1: identify object and their way of repr
following predicates are define on table
ontable (x) : block x is placed on table.
ontop (x,y) : block x is on top of block y
clear (x) : block x has clear top.
Holding (x) : Robot arm is holding block x.
Empty : Robot arm is empty.

Step 2: define initial state and goal state
Initial State: ontable (A) ∧ ontop(A) ∧ clear (C) ∧ ontable (B) ∧ clear(B) ∧ Empty

Goal state : ontable(A) ∧ ontop(B,A) ∧ ontop(C,B) ∧ clear(C) ∧ Empty

Step 3 : identify the operators with their pre conditions and effect.

- 1 Action : Pickup(x) : (Pickup block x from table)
  precondition : Ontable(x) ∧ clear(x) ∧ Empty
  Effect : Holding(x)

- 2 Action : Putdown(x) : (Putdown block x on table)
  precondition : Holding(x)
  effect : Ontable(x) ∧ clearc(x) ∧ Empty

- 3 Action : Unstack(x,y) : (Pickup block x from top of y)
  precondition : ontop(x,y) ∧ cloar(x) ∧ empty
  effect : Holding(x) ∧ clear(y)

- 4 Action : Stack(x,y) : [Putdown block x ~~on~~ block
  precondition : Holding(x) ∧ cloor(y)
  effect : ontop(x,y) ∧ cloor(x) ∧ Empty

Other than above action every plan must consist of 2 dumy action start & finish

- 5 Action : START
  precondition : Nil
  effect : Initial state

- 6 : Finish
  precondition : Goal state
  effect : Nil

Step 4: plan (Strict order plan)

START

[ontable(A) ∧ ontop(C,A) ∧ clear(C) ∧ Empty ∧ ontable(B) ∧ clear(B)]

↓

Unstack(C,A)

[ontable(A) ∧ ontable(B) ∧ clear(A) ∧ holding(C) ∧ clear(B)]

↓

Putdown(C)

[ontable(A) ∧ clear(A) ∧ ontable(B) ∧ clear(C) ∧ ontable(C) ∧ clear(C) ∧ empty]

↓

Pickup(B)

[ontable(A) ∧ clear(A) ∧ ontable(C) ∧ clear(C) ∧ holding(B)]

↓

Stack(B,A)

[ontable(A) ∧ ontop(B,A) ∧ clear(B) ∧ empty ∧ ontable(C) ∧ clear(C)]

↓

Example 2: Provide a plan for "Put on pair of Shoe"

Step 1: identify object and their way of repres
we need following propositions
1. left sock.on
2. Right Sock.on
3. left shoe.on
4. Right shoe. on

step 2: define initial state & goal state
Initial state: Nil
Goal state: left shoe.on ∧ right shoe.on

Step 3: identify operators with precondition & effec

1 Action 1: Put on .left Sock
Precondition: Nil
Effect: left Sock.on

2 Action: Put.on.Right sock
Precondition: Nil
effect: right sock.on

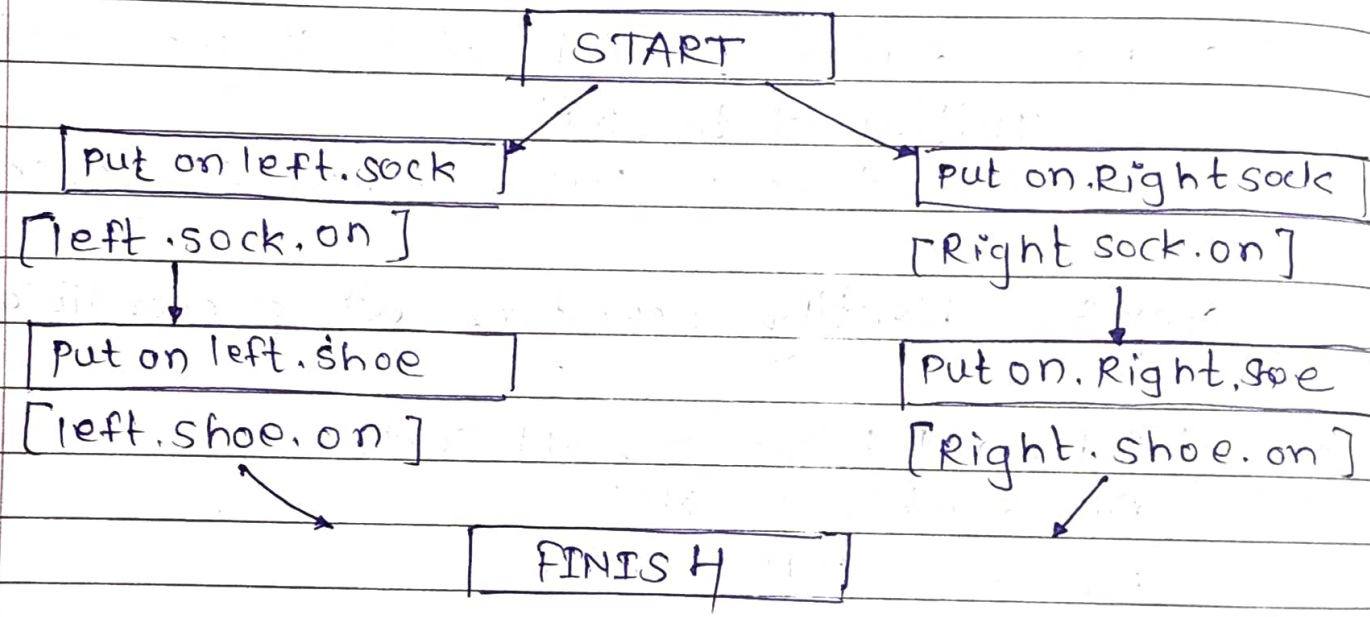
3 Action: Put.on.left.shoe
precondition: left Sock.on
effect: left shoe.on

4 Action: Put on Right.shoe
precondition: Right sock.on
effect: Right shoe.on

5 Action: START
precondition: Nil
effect: Initial state

6 Action: finish
precondition: Goal state
effect: Nil

step 4: Plan (Partial order plan)



Types of plan:

1) Strict order plan
2) partial order plan
3) Hierarchical plan
4) Planning with time scheduling

5M write a short note on plan? (steps & components)

- what is plan
- steps of plan (step1, step2...)
- Component of plan:
  1) every plan consist of (sequence of actions), ordering constraints

2) The ordering constraint between Action A & Action B is given as A < B (A before B) It means action A must be taken any time before Action B.
causal links : coupled link
3) The causal link between action A & action B is given as $A \xrightarrow{P} B$ which means action A achieves precondition for action B. and action A must be taken immediately before action B.
4) List of open preconditions :
The open preconditions are those precondn which are not achieved by any action in plan

Example of Resolution

FOL
I) The law says that it is a crime for a american to sell weapons to a hostile nation. the county nono is a enemy of america and has some missile. and all of its missile where sold to it by Cournel west who is an american
Q ⇒ Prove that "west is criminal"

→ following are the facts
1) It is crime for a america to sell weapons to a hostile nation.
2) The county nono is a enemy of america.
5) Nono has some missile
3) west is a american
4) All the missles that nono has are sold it by cournel west
6) All missles are weapons

7) Enemy of america is a hostile.

Predicate logic :

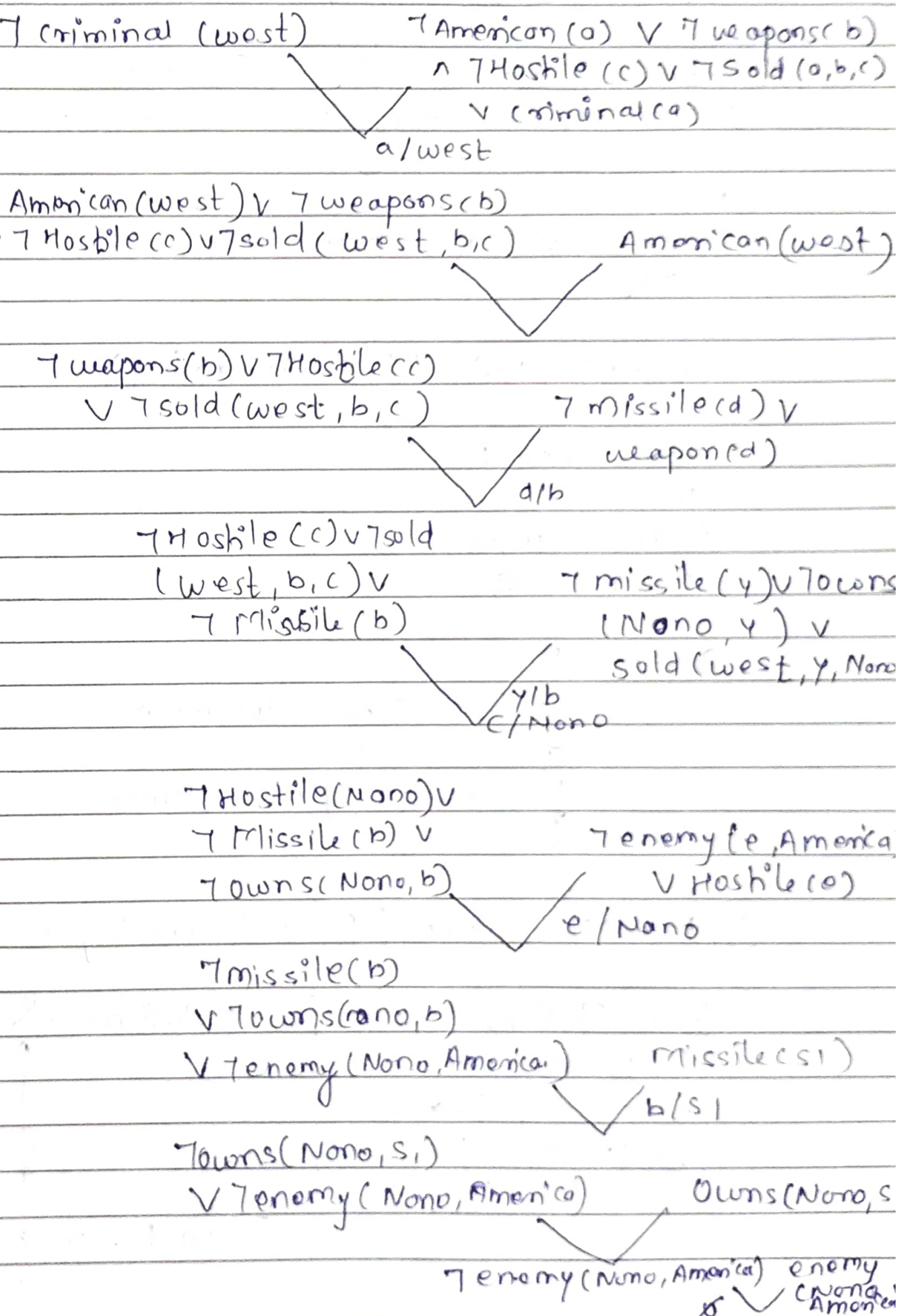
1) American (west)
2) Enemy (Nono, America)
3) $\exists x$: Missile(x) $\wedge$ Owns(Nono, x)
4) $\forall y$: Missile(y) $\wedge$ Owns(Nono, y)
   $\rightarrow$ sold (west, y, Nono)
5) $\forall a : \forall b : \forall c$: America (a) $\wedge$ weapons(b) $\wedge$ Hostile(c)
   $\wedge$ sold(a, b, c) $\rightarrow$ criminal (a)
6) $\forall d$: Missile (d) $\rightarrow$ weapons(d)
7) $\forall e$: enemy (e, America) $\rightarrow$ Hostile(e)

Predicate to CNF :

1) American (west)
2) Enemy (Nono, America)
3) ~~$\exists x$~~ : Missile (S1)
   Owns (Nono, S1)
4) $\neg$ missile (u) $\vee$ $\neg$ Owns (Nono, u) $\vee$ sold (west, y, Nono)
5) $\neg$ American(a) $\vee$ $\neg$ weapon (b) $\vee$ $\neg$ Hostile (c) $\vee$ $\neg$ sold (a, b, c) $\vee$ criminal (a)
6) $\neg$ missile (d) $\vee$ weapons (d)
7) $\neg$ Enemy (e, America) $\vee$ Hostile(e)

prove that criminal (west) is true Negate:
$\neg$ criminal (west)

Resolution


¬ criminal (west)
¬ American (a) ∨ ¬ weapons (b)
∧ ¬ Hostile (c) ∨ ¬ sold (a, b, c)
∨ criminal (a)
a/west
¬ American (west) ∨ ¬ weapons (b)
∨ ¬ Hostile (c) ∨ ¬ sold (west, b, c)
American (west)
¬ weapons (b) ∨ ¬ Hostile (c)
∨ ¬ sold (west, b, c)
¬ missile (d) ∨
weapon (d)
d/b
¬ Hostile (c) ∨ ¬ sold
(west, b, c) ∨
¬ missile (b)
¬ missile (y) ∨ ¬ owns
(Nono, y) ∨
sold (west, y, Nono
y/b
c/Nono
¬ Hostile (Nono) ∨
¬ missile (b) ∨
¬ owns (Nono, b)
¬ enemy (e, America
∨ Hostile (e)
e/Nono
¬ missile (b)
∨ ¬ owns (nono, b)
∨ ¬ enemy (Nono, America)
missile (s1)
b/s1
¬ owns (Nono, s1)
∨ ¬ enemy (Nono, America)
Owns (Nono, s
¬ enemy (Nono, America)
enemy
(Nono,
Americ

Reasoning / chaining

1) Forword Reasoning : Given → conclusion
2) Backword Reasoning : Conclusion ← Given

P

P → Q

Q → R

P.T: R is true.

Inference Rule (F/w Reasoning)

Modus Ponen :

Given P

Rule P → Q

Infer Q

Given P

↓ from (2) by modus ponen

Q

↓ from (3) by modus Ponen

Conclusion R

Inference Rule (B/w Reasoning)

Modus Tollens :

To prove : Q

Given : P → Q

first prove : P

Conclusion → R

↑ from 3 by modus Tollens

Q

↑ from 2 by modus Tollens

Given T → P

consider the following facts

1) If hot and smoky then fire
2) If alarm beeps then smoky
3) If fire then switch on sprinkler
4) alarm beeps
5) It is hot.

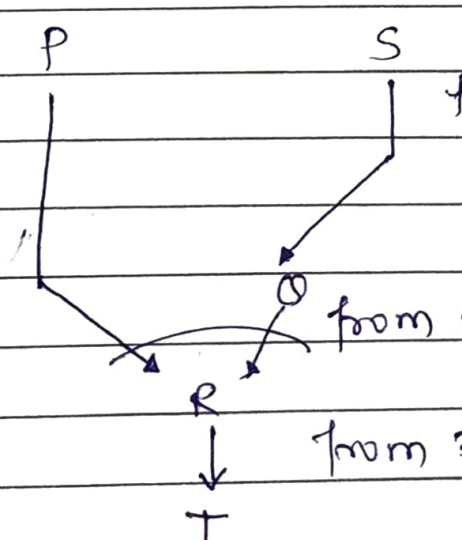
P.T Sprinkler is on using forward & backward reasoning.

→ convert above fact to propositional logic

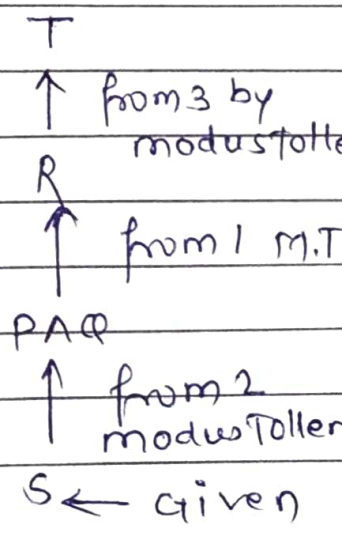
It is hot → H P
It is smoky → S Q
It is fire → F R
It is Alarm → A S
Switch on sprinkler → P T

1) $P \wedge Q \rightarrow R$
2) $S \rightarrow Q$
3) $R \rightarrow T$
4) S
5) P

Forward Reasoning



Backward Reasoning



Prove that west is criminal using f/w & B/w Reasoning.

Module 6

Steps in NLP - 10 VIMP

Healthcare, Retail, Banking Case study