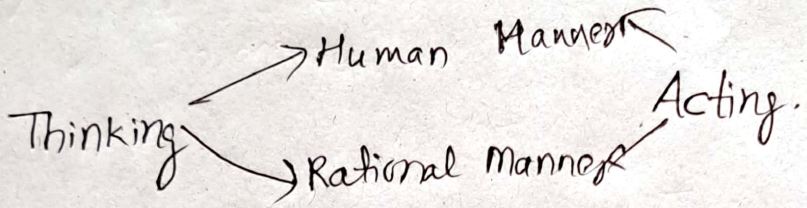
Artificial Intelligence

AI is making machines that can think & do like humans can do.

1956 ⇒ Dart Month Workshop.



		T	A
Human	H	✓	✓
Robot	R	✓	✓

Turing Test :-

Human ← Chatbox → Machine

Intelligent Machine :- That can think like Human.

ANN ⇒ Artificial Neural Network

Binary Logic :- 0/1 T/F

Fuzzy Logic :- between 0 to 1 (conditional)

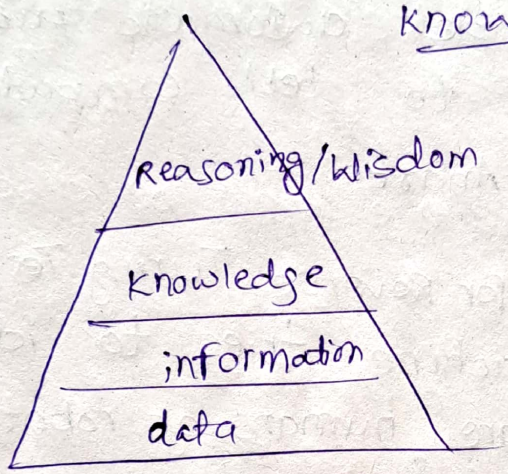
With same input each time, output will improve each time due to intelligent program.

- Magic Square / Tic-Tac-Toe / N-Queen,

- Turing Test :- A machine acting like Human.

(Text related)
- → NLP
- → Knowledge Representation, (★★)
- → Automated Reasoning (uses ∧, ∨, ⊕, ¬, Truth Table)
- → Machine Learning.

Knowledge Pyramid



DIKW

- Ontological tree is used for representation of knowledge.

x = Today is Thursday — True

y = This notebook is laptop — False

z = This sentence is True. — cannot be predicted

For reference :- Paradox R

Machine domain :- Remove ambiguous statements
:- Keep only statements with Truth values

- Russell's Paradox!!

Total turing Test

- Text, Voice, speech, video
- + 4 skills
  - NLP
  - Knowledge Representation
  - Automated Reasoning
  - Machine Learning

CAPTCHA :- Complete automatic public Turing Test to tell computers & Humains Apart.

- used for Reverse Turing Test where Machine tries to identify whether we are humans or robots

---

Summary :- AI is not only reverse Turing Test. It is doing tasks that human can do, but not mimicking them.

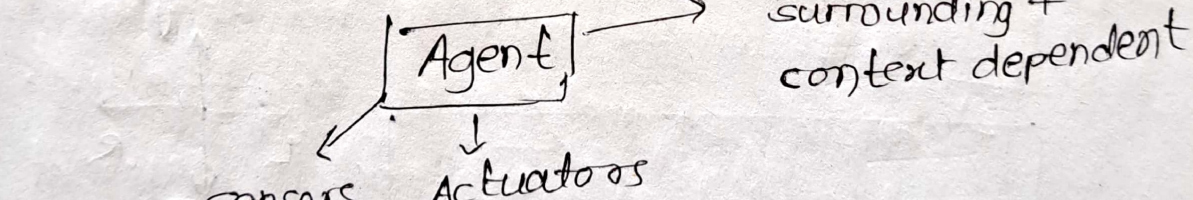
Acting Humanly

Thinking Humanly

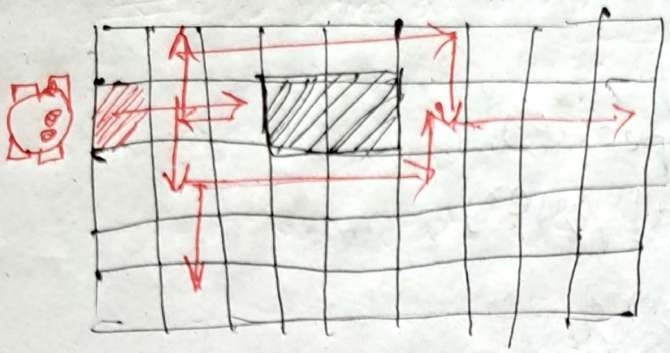
- self introspection
- psychological query
- brain mapping.

Hypothesis :- A program or machine that can perceive from its environment is called Agents. (Intelligent Agents - matrix)

- Agent observes its environment. (Agent Smith).
- Agent is possibly a robot.
- Robot/Agent will also have computer vision.
  - sensors / Actuators.



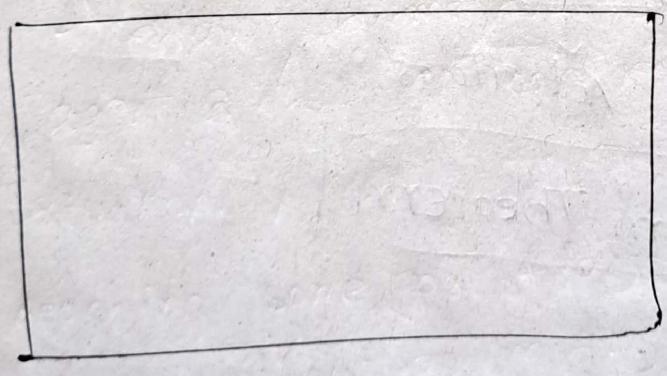
Chinese Room Theorem

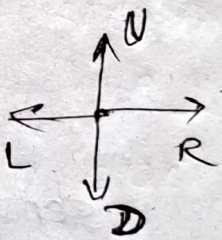


- city block distance (manhatten dist)
- chessboard distance (manhatten distance)
- Euclidean Distance
  - ★ we teach robot to move in steps
  - ★ hence distance must be discrete whole nos

- Agent will record its moves.
- sequence will be called as percept sequence

set of actions made by agent





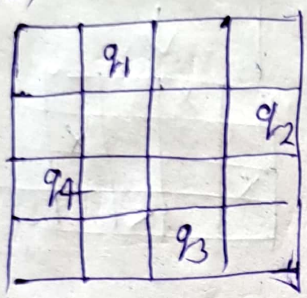
Percept sequence :- moves made by agent in a sequence

:- Percept sequence will be used to Learn

4-Queens problem

Water jug problem :- 3 li, 4 li, ⟷ 4 li

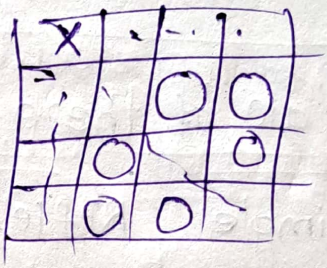
(0,0)



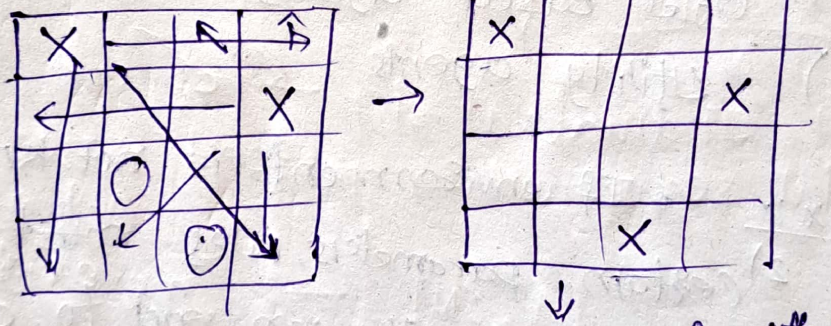
(3,3)

Solution

- give in form of steps
- place first queen at (0,0).
- mention rules to agent



- Place in any of the circles



↓

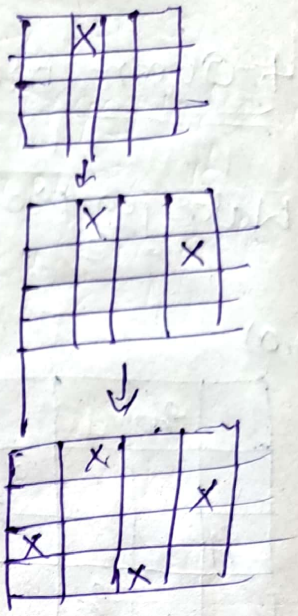
no space for 4th queen

↳ Rollback

Steps

place at (0,0)
find next position.
place at (1,3)
place at ...
.
.
.
Rollback to previous & previous steps if not possible.

Again
- place at (0,1)
- place at (1,3)
- place at (2,0)
- place at (3,2)



Agents are of these types:-

1) simple Reflex Agents ex:- automatic car.
2) Model based ~~Agents~~ reflex agent.
3) Goal based agents.
4) Utility agents

ex:- • If environment is not known.
2) certain parameters, rules must be given to agent & that it can react to current situation.

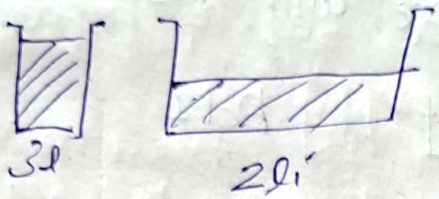
3) • Reaching some goal & target without optimizing parameters.

4) Goal + extra condition (optimisation)

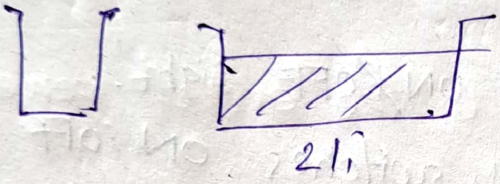
Water Jug problem

Sol 1

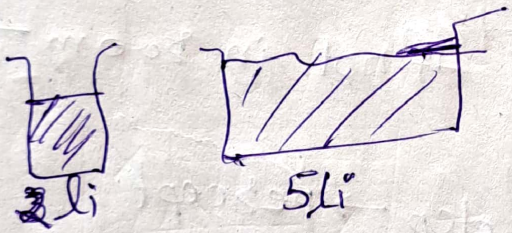
- Fill 5 li vessel.
- Pour 5li in 3li



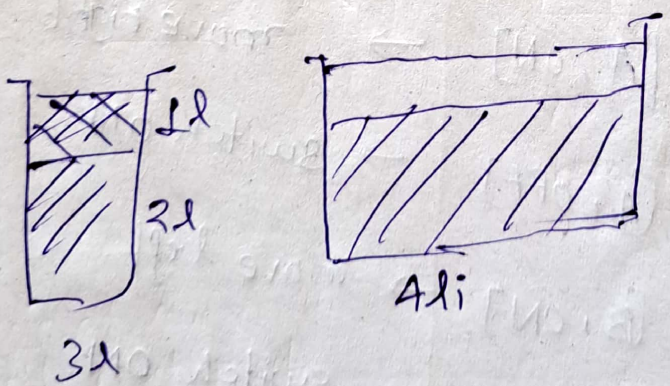
- Empty 3l.



- Pour in 3l & Fully fill 5li



- Pour 5li in 3li



Sol 2

	5l	3l
step1	0	0
step2	0	3
step3	3	0
step4	5	0
step5	2	3
step6	2	0
step7	0	2
step8	5	2
step9	4	3

Percept sequence of an Agent → sensor, → actuators

- As agent does its steps, it remember it.
- Percept sequence will result in action

[ ] Act1

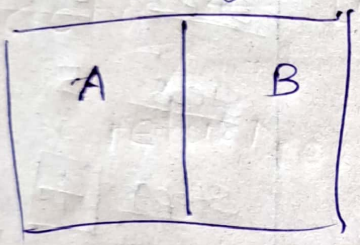
experience

[ ] Act2.

---

ex:- Agent to switch ON/OFF light.

- An agent which switches ON/OFF light.



If we are in room → ON

If not in room → OFF

Individual step → percept

Total sequence → percept sequence.

percept	Action
[A, ON]	→ move right
[A, OFF]	→ switch ON
[B, ON]	→ move left.
[B, OFF]	→ switch ON.

∴ percept sequence:

[A, ON], [B, OFF], [B, ON], [A, ON], [B, ON], etc.

*Exam question

- Read about environment
- Find valid actions using sufficient hypothesis

Agent = Architecture + Program

- AI: modern approach Deepak adani

Books.

- Peter Russel :- AI : ~~000~~
  & Norwick ✓
- Rich & Kevin night (1991)
- AI for dummies (Head First)

After unboxing agent:

- If stores a step a first step in its table.
- we can use any ~~data~~ structure for table.

~~Read life exam~~

Real life Examples :-

Function { Table Driven Ag (percept) returns an action.

persistent : percepts - a seq. of percepts, initially empty.

table - a table of actions, indexed by percept sequence

append percept to the end of percepts.

action ← Look Up (percepts, table).

return action.

}

End

Camel & banana problem

Person
Camel

1000 km

Dest.

3000 bananas

1 banana / km

- If person moves with camel, after 1 km, 1 banana req.
- Camel at most 1000 bananas.

Fun Simple Ref Ag (percept)

persistent rules - a set of if-then rules

State ← Interpret Input (percept)

rule ← Rule match. (state, rules)

action ← rule. Action.

return action

End

Q] Agents learn methods——?

reinforcement learning (RL)

ans- rewards & punishments are awarded.

Problem Solving

Goal based Agent

State Space Search

P

Sol.

Stop Goal state.

Einstein - we cannot solve problem with the same thinking that created them.

- State space search questions will be solved using State Space search Tree.
  - These methods are known as uninformed search
  - We try connecting initial state to final state

Problem solving will have 4 things:-

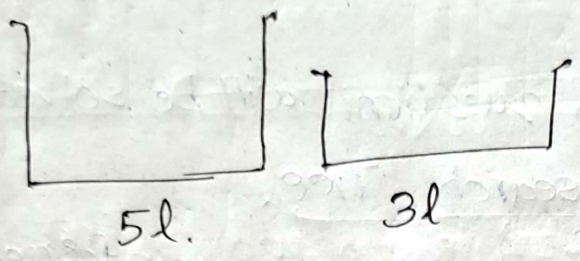
1) Define the problem
2) Analyze the problem.
3) Isolate & represent the knowledge.
4) Choose the best Tech.

- ~~define~~ problem ~~as statement~~ ~~analysis~~

5 components to define problem

- initial state
- description of actions
- transition model.
- Goal Test
- Path cost estimation.

ex:- Water-jug problem

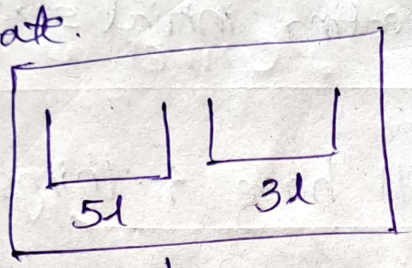


Assumptions

- infinite water
- u can waste water
- No measurement marks on container.

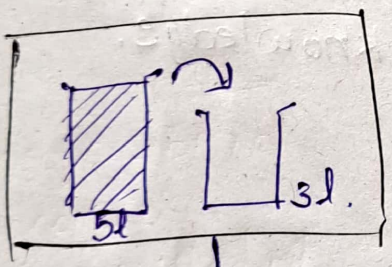
ans:-

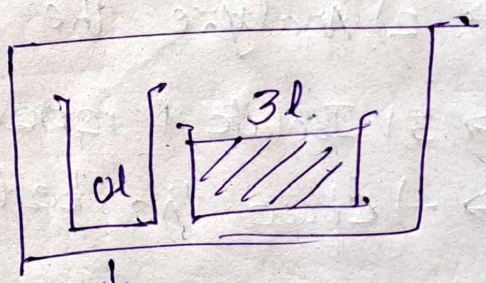
Initial state.

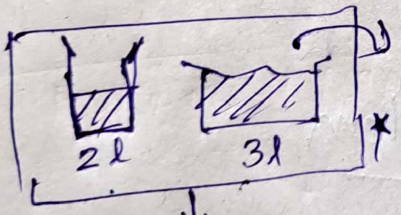


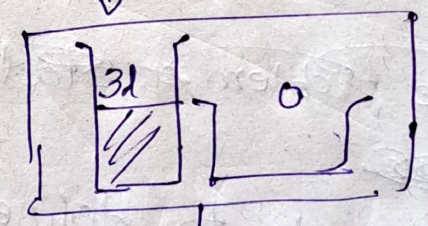
a

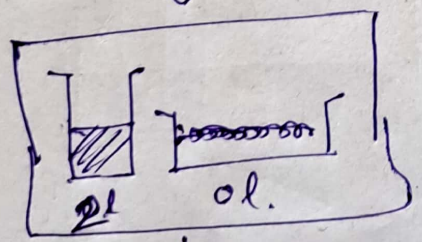
b

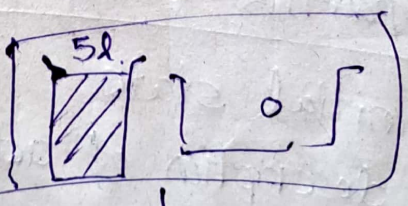


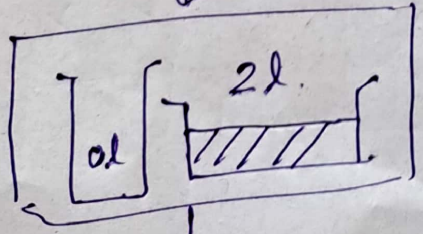


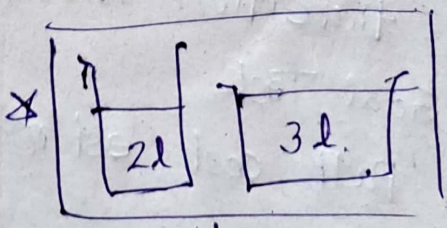


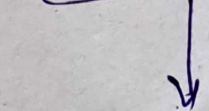


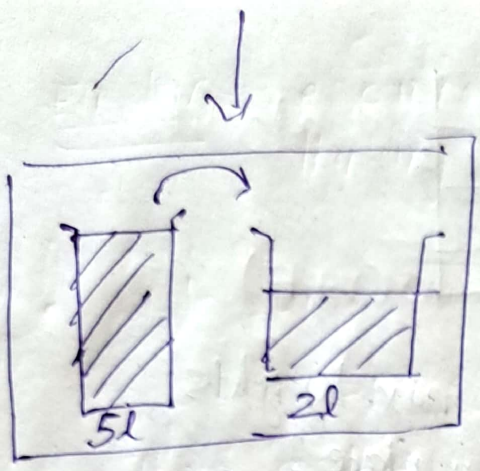


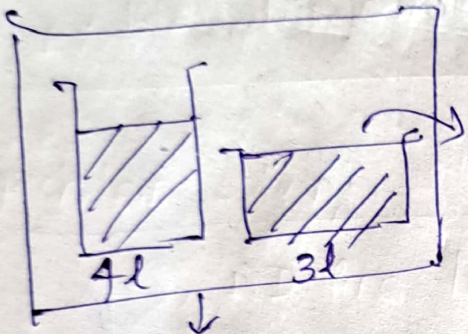


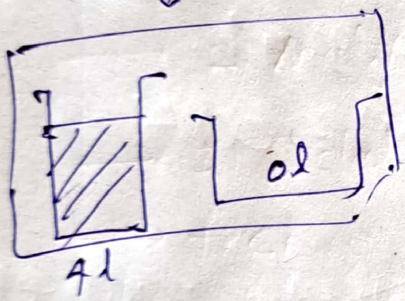




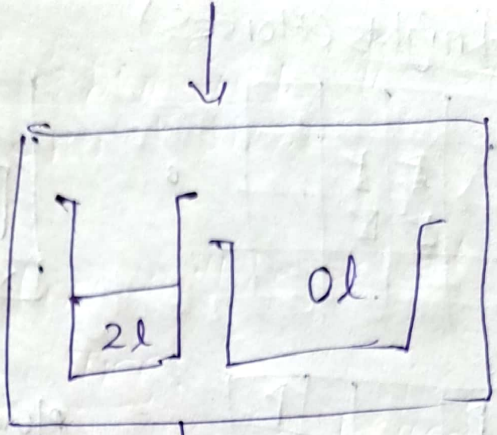


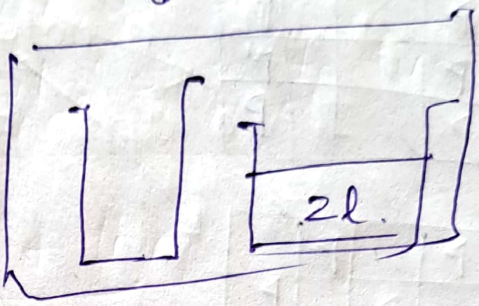


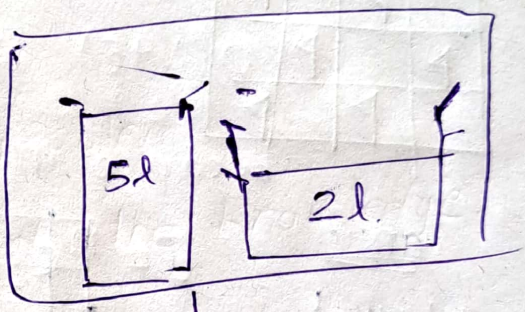


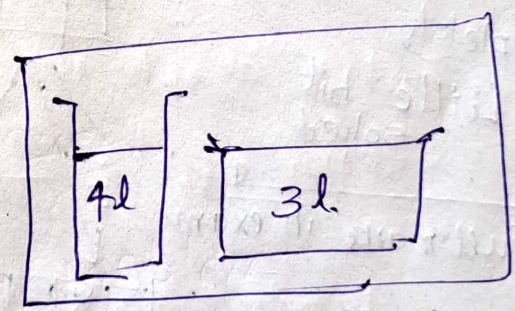


7 steps.





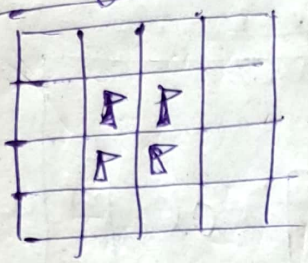


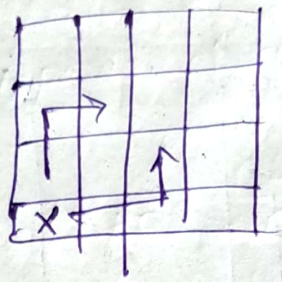


8 steps.

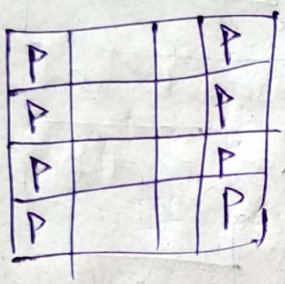
path a > path b (less steps)
(less water wastage)

4X 4knights (Horses)



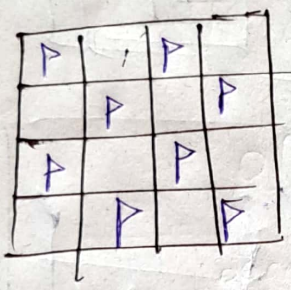


or



How many max. Knights can u place?

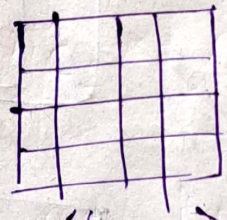
ans: 8

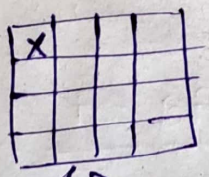


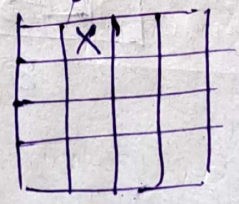
ans:

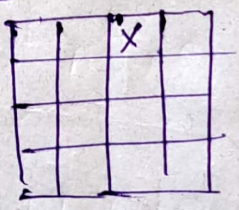
- Solve 1 branch completely
- Rest Little bit solved

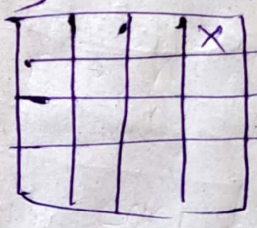
Full marks in exam

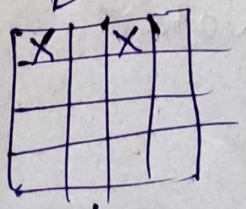


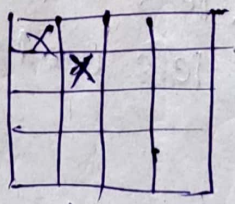


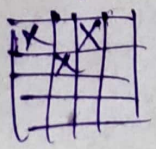












Q] Camel Banana problem :-

3000 km, 1 banana per km.

Russel-Norwick Book

Formulate - Search - Execute Model :-

- These models are tested on completeness & optimality (Time & space).
  (Majority Time complexity)

```
Search ( )
{
    OPEN ← {START};
    while OPEN != Empty.
    do Pick a node 'n' ∈ OPEN
        OPEN ← OPEN \ {n}.
    if Goal Test (n) = True Return n.
    else OPEN ← OPEN ∪ Move Gen (n)

return FAIL.
}
```

* OPEN is bag data structure (like set).

MoveGen :- means generate child nodes.

Sir will ask Time complexity of any algorithm in Exam.

ex:- Time complexity of BFS:- $O(V+E)$.

Q) Graph with n nodes $G(V,E)$.

If we apply BFS, what will be Time complexity.

ans:-

$O(V+E)$ or $O(n^2)$

because in complete graph,

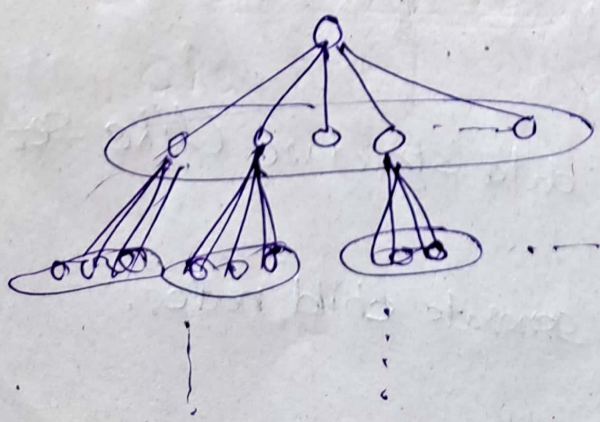
$$E = {}^nC_2 = \frac{n(n+1)}{2} = O(n^2)$$

But a graph may have self edges, multi-edges

Hence; E — $f(n)$ we don't know relation

$E \approx n^2$ or n^3, or n^e, or $n \log n$, etc?

For BFS, Take 'b' branches at each layer and 'd' depth of tree.



$\therefore$ Time complexity $= O(b^d)$

Q] Is BFS complete?

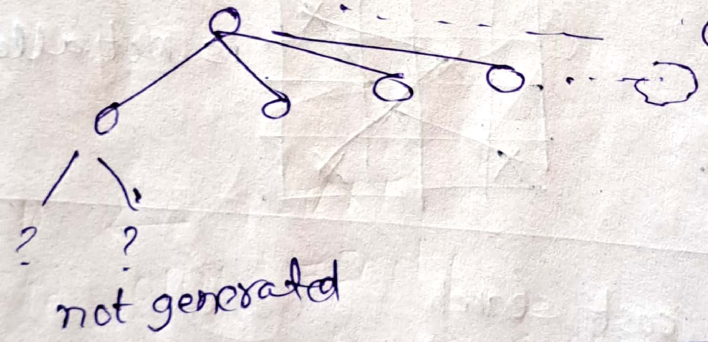
ans:- Yes, it will give whole solutions if it exists

∵ Algorithm returns if goal state Reached.

- If goal does not exist, it explores all paths.

but if b or d is infinite.
Then solution will never be found.

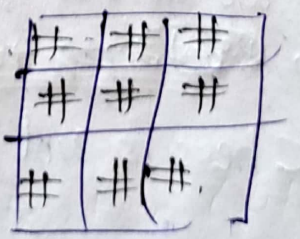
Note:- ~~We will not generate all~~
BFS theoritically generates each node & moves down.



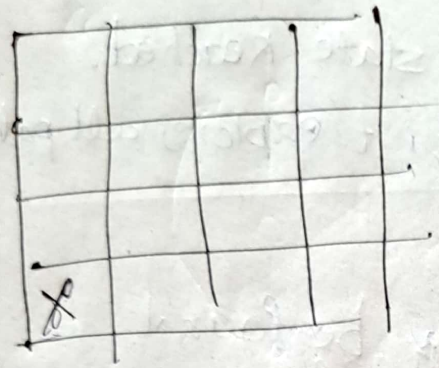
- We find a instant solution, that is not generalised. Humans do like this.
- But machines generalise it for any 'n'.

search (Tic Tac Toe)^2

Reverse Tic Toc Toe.



Knight's move / Tour



ver 1

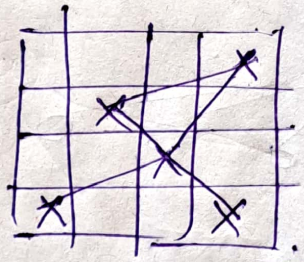
ver 2

- cover whole grid, with Knight's step.

cannot be solved for ~~[illegible]~~, 2x2, 3x3.

ver 1 • Knight has to visit whole board, without repeating any cell.

ver 2 The lines should not intersect.



is not allowed

- uniform cost search
- Depth first search
- Depth bounded / Limited search
- Best fit search.

Facts :- we do not know solution.

~~Problems in chess~~ :-

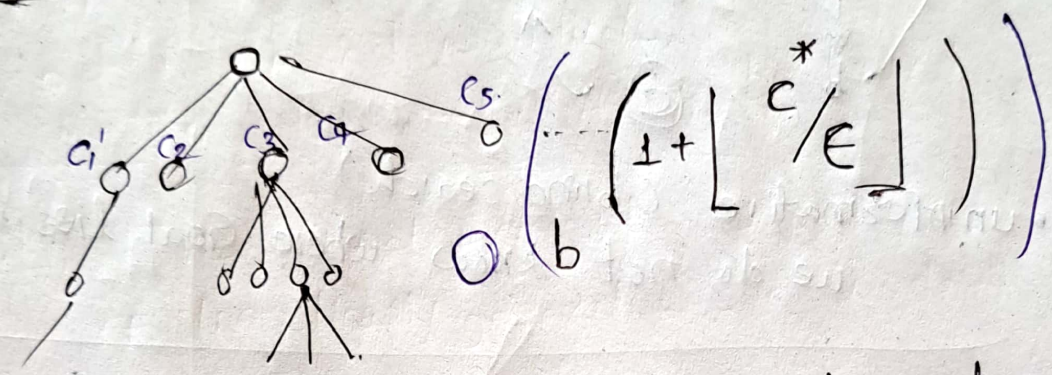
Chess is not solved problem.

Uniform cost search :-

State space search

BFS :- $O(b^d)$

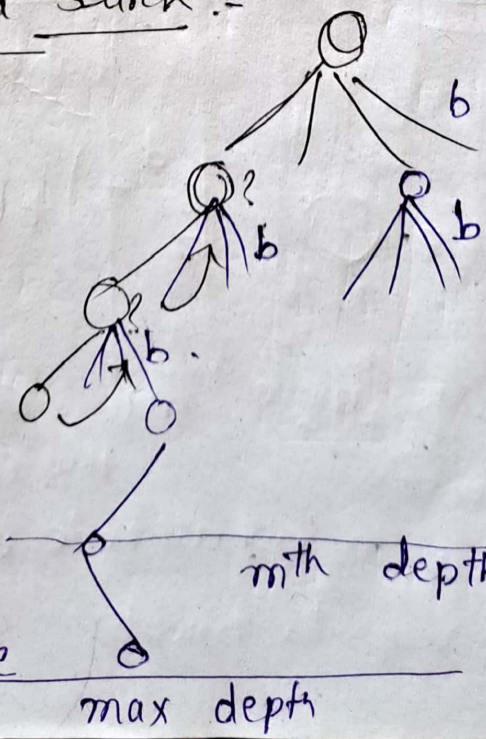
Dijkstra's Algo :- with cost optimized comparison



- BFS is specific case of uniform cost search
- If cost of each search is Equal, Dijkstra's Algo becomes $O(b^{d+1})$

c^* - optimum cost.

Goal Test based search :-



Time complexity :

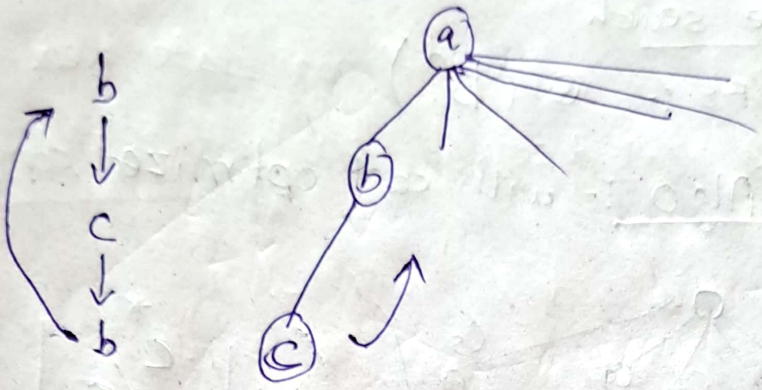
$O(b^m)$

worst case

m^{th} depth :- b^m branches.

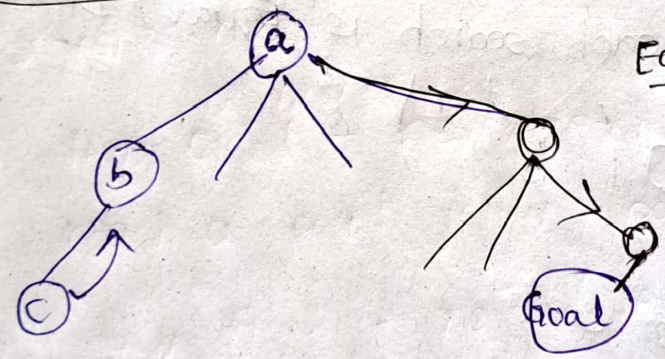
worst case
max depth :- b^d branches.

DFS can have infinite loop problem



- uninformative (blind search)
  we do not know where Goal lies in Tree.

Informed search (Heuristics search) :-

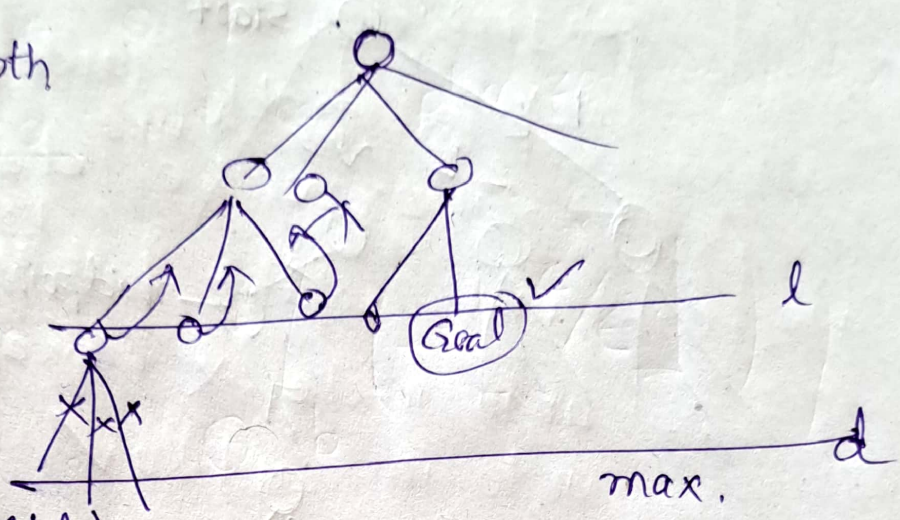


Easy.

ex:- Google maps changes Route based on Traffic.

Depth Limited search :-

If goal is at depth 'l'.



- Go till 'l' depth
- back track after it.

- Time complexity :-

$O(b^l)$

- Space complexity :- $O(bl)$

If $l=d$ or $l=\infty$, then DLS = DFS.

Note :- Space complexity in DFS :- $O(bm)$

Iterative Deepening :-

- Apply Depth Limited search sequentially

Let $d=0$, find all nodes.
$d=1$, explore nodes, Look for Goal.
$d=2$, explore nodes, Look for Goal
:
If Goal found, backtrack all nodes

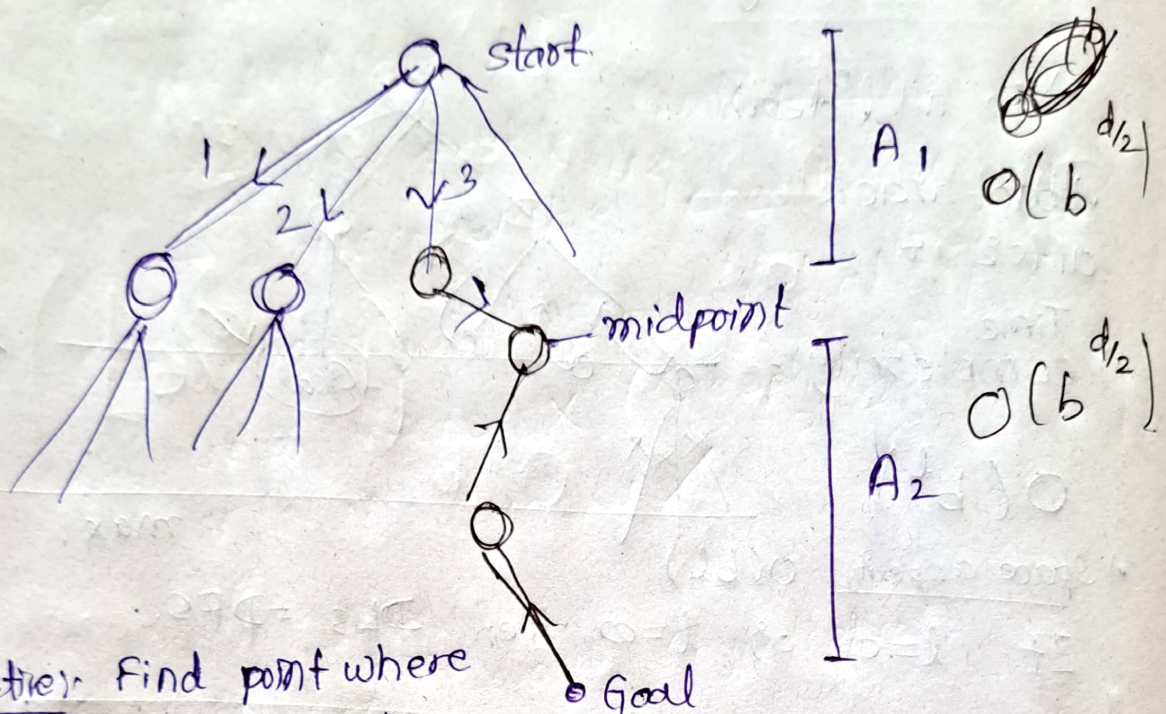
This is restricted form of DFS & BFS.

:

If $d=d$ ⇒ Depth first search.

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

Bi-directional search (BDS)



objective: Find point where both searches meet

•) Start from beginning & never reach the Goal, but reach a point joining both searches

•) Start from Goal & never reach beginning but reach the point joining both searches

Time complexity will be halved $O(b^{d/2})$

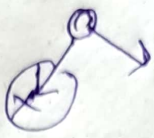
Time complexity = $O(b^{d/2}) + O(b^{d/2})$

problem:- How to move from Goal to its predecessors. (H.W (saturday Lab - monday Timetable))

You will have to visit all nodes once to remember & map the parents

	complete	optimal
BFS:-	✓	✓
	traverse all nodes & find Goal	Yes, if goal is at less depth, we will have more to more depth for other goals. We will find first goal & return
Uniform cost search :-	✓	✓
DFS :-	✗	✗
	• if Loop trap. never find Goal	• If goal not found, what about optimisation.
DLS :-	✗	✓
BDS :-	✓	✓
Iterative deepening	✓	✓

Can solution steps be undone?



(ignore solution)

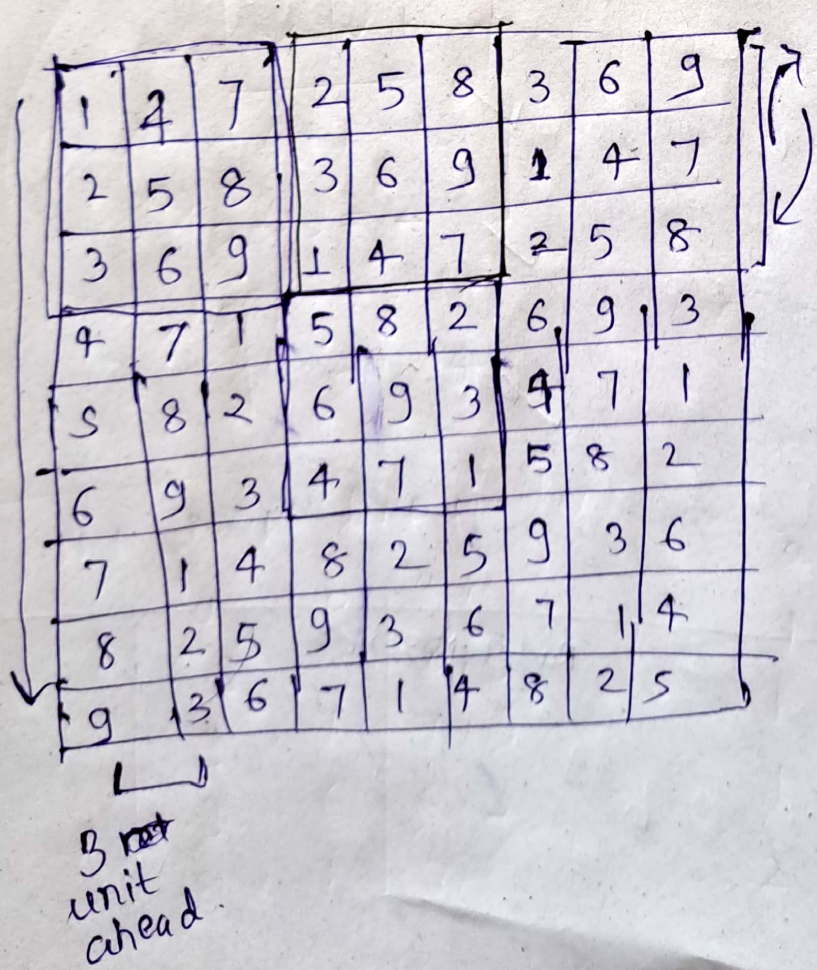
3 possibilities

1) ignorable
2) recoverable (undo possible)
3) irrecoverable (chess)

We can undo solution, but not always

ex: undo in chess X (Gameplaying.)
undo in Sudoku ✓

Problems → Toy world (hypothetical)
→ Real world (Robotic movement)



Exam How magic square can solve Tic-Toe?

ans:- Heuristics:-

certain problems can be transformed from one domain to another.

- Problem with Tree algo for Tic-Tac-Toe:-

combinatorial Explosion.

Heuristic functions will help in this.

magic square

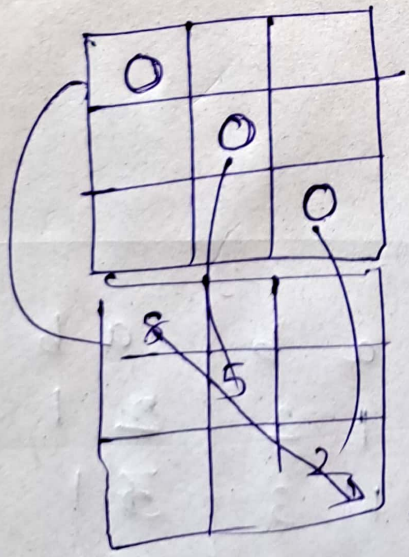
1	2	3
4	5	6
7	8	9

location grid

00	01	02
10	11	12
20	21	22

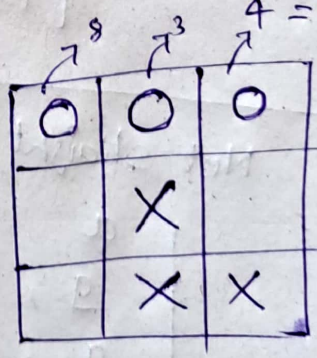
Transformed magic square :-

8	3	4
1	5	9
6	7	2



any winning position will give sum 15

ex:-



a) First move - decide nothing
b) second move - some winning position.
c) 3rd move - winning position becomes clear.
ie if (a+b+c) =15 ✓ win.

if (15 − (a+b) ≤ 0 or 15 − (a+b) > 9)

ex:-

0		X
X		0

a, b are not in straight line.

8 + 9 = 17

15 − 17 = −2

8	3	4
1	5	9
6	7	2

1 + 4 = 5

15 − 5 = 10 > 9.

ex:-

else a, b in straight line

0	0	
X	X	

6 + 7 = 13

15 − 13 = 2

8 + 3 = 11

15 − 11 = 4.

Not straight line positions:-

a	b	a	b
9	8	8	9
9	7	7	9
9	6	6	9
8	7	7	8

a	b	a	b
1	2	2	1
1	3	3	1
1	4	4	1
2	3	3	2

Lets play Game

O		
	X	
O		

1 2 3 4 5

O :- 8, 6, ☐

X :- 5, , ,

If both players use same strategy, They will also block other one.

8+6+1 = 15.

X will block O.

O		
X	X	
O		

O will block X

1+5+9 = 15

O		
X	X	O
O		

X will try another combination.

O	X	
X	X	O
O		

O will block X.

O	X	
X	X	O
O	O	

O	X	
X	X	O
O	O	X

X will block O.

O	X	O
X	X	O
O	O	X

1 place remaining,
no willing position
place O.

Dual. *we will use magic square as Heuristic func.

Heuristic - extra info will be given.

Heuristic function will be given to us.

Best first search

↓

A^*

↓ utilises this.

$f(n) = g(n) + h(n)$ → Heuristic function

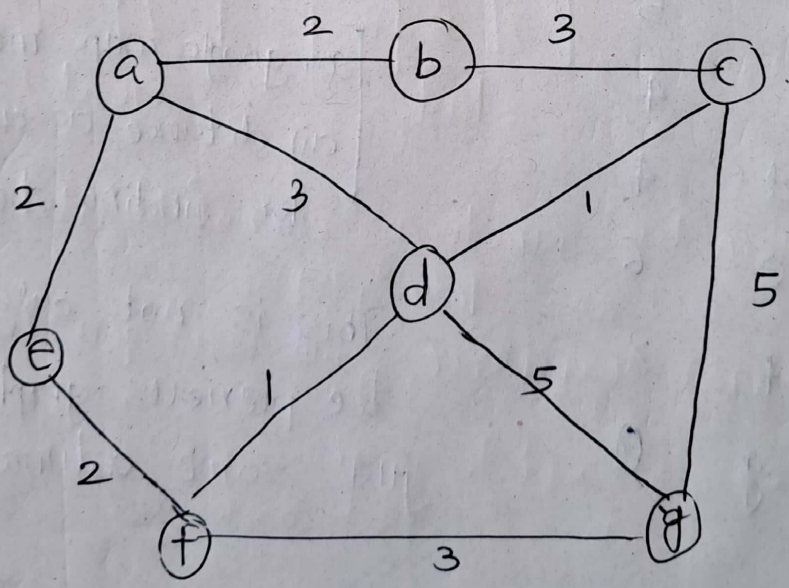
↳ cost fun

- Heuristic is better than blind search

- They are trying to estimate 'distance' of goal.

Heuristic:- an evaluation function used to estimate the cost of reach goal state.

Find shortest path



- Dijkstra will give only 1 path.
- Heuristics is not specified.

$g(n)$ | $g(e) = 2$
$g(a) = ?$
$g(b) = 2$
$g(d) = 5$.

cost function

What is heuristic func here?

We had not provided,
we will provide now:-

estimated distance of goal from these node

[this is not exact distance from goal,
else $h(a)$ will be the answer.]

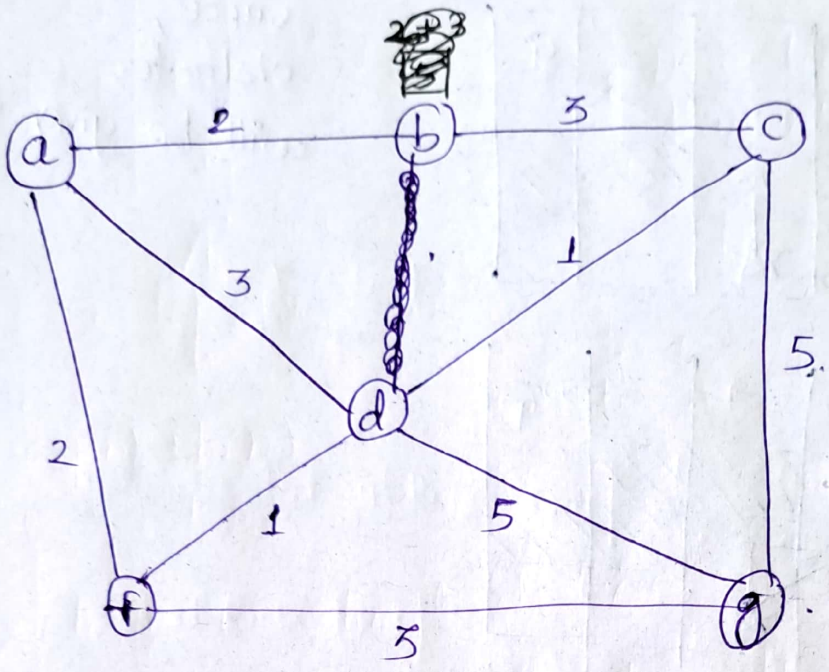
a	7
b	5
c	4
d	4
e	6
f	3
g	0

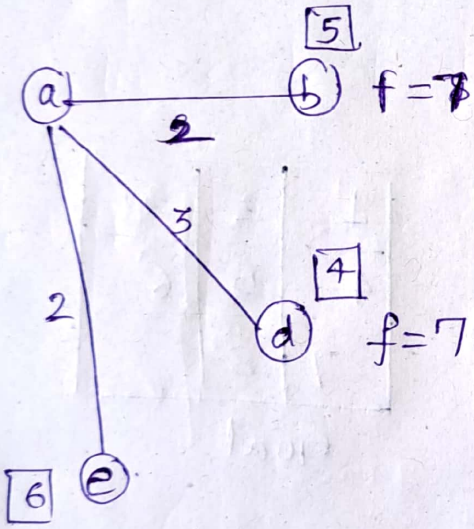
This can be treated as air distance / Euclidean dist, etc.

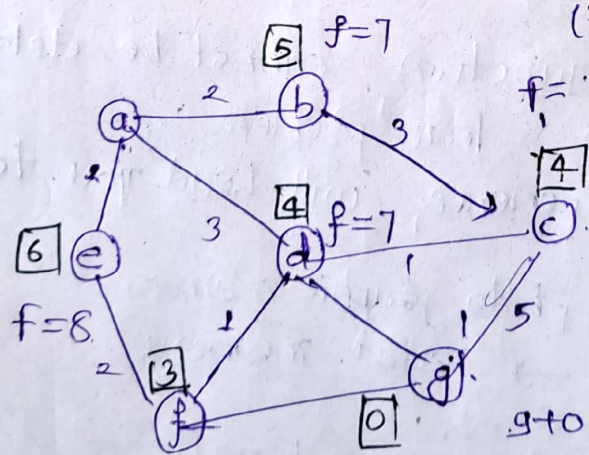
[so, google maps may use air distance as an heuristic for finding shortest path.]

This is not calculated from the previous graph, it is just some estimate.

we will use $f(n) = g(n) + h(n)$







(2+3) 4
$f = g + h = 9$ ✗

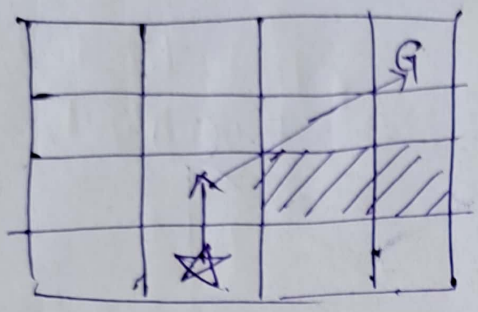
$f_2 = g + h = 8$ ✓
(3+1) (4)

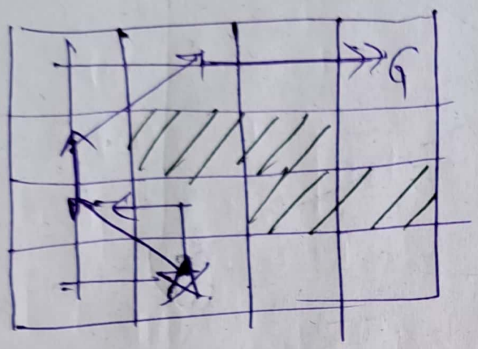
$f = g + h = 7$
↙ ↓
(2+2) 3

g+0 = 9 ✗
8+0 = 8 ✗
7+0 = 7 ✓

A* will be asked in exam

Maze problem
grid,
source,
obstacles
will be given to you.





Heuristics For 8-puzzle

2	1	4
3		6
7	5	8

start

1	2	3
4	5	6
7	8	

Goal.

- Heuristic function cannot be determined so from problem.
- only Experience will Lead you to it.

- Humans → photo graphic memory
- Computer → serial memory.

h(n) → no. of displaced cells.

~~Quadratic~~ Square roots :-

$\sqrt{2}$

pen-paper:-

	x	y
x	$2.\bar{0}0\ \bar{0}0\ \bar{0}0$	
	x^2	
y	00	
	y^2	
	00 ...	
	:	

In computer programs, we use binary-search (or) bisections.

- use quadratic equation:-

$ax^2+bx+c=0$ $\qquad x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$.

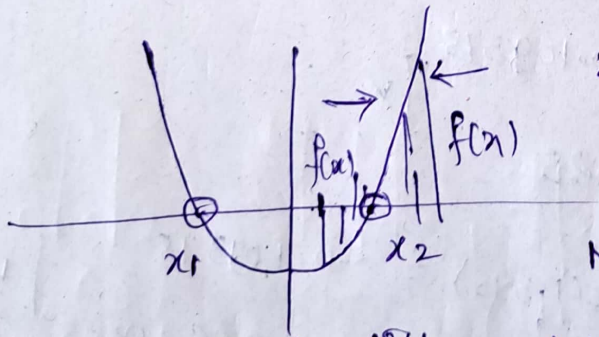
not like this $\begin{cases} x^2-2=0 \\ x^2=2 \end{cases}$ | $a=1,\ b=0,\ c=-2$

$\therefore x=\frac{\pm\sqrt{-4.1.(-2)}}{2}$

$x=\frac{\pm\sqrt{8}}{2}$.

$x=\frac{\pm 2\sqrt{2}}{2}=\pm\sqrt{2}$

draw graph of $x^2-2=0$.

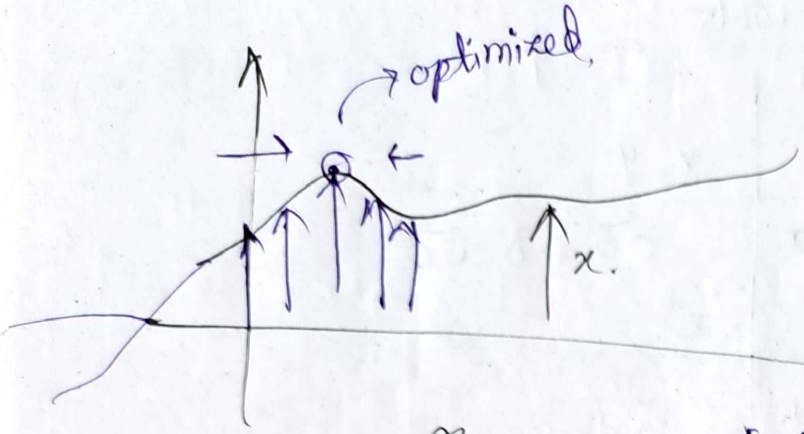


x_1, x_2 are roots.

move untill $f(x)=0$.

Newton-Raphson method.

we calculate till certain accuracy.



Hill climbing algo.

$x \longrightarrow$ $\quad f(x) = x^2 - 2$

$f(0) = -2$

$f(1) = -1$] root will be between them

$f(2) = +2$

$f(1) = -1$] root will be here

$\therefore f(1.5) = +0.25$

$f(2) = +2$

$f(1) = -1$] ✗

$f(1.25) = -0.4375$] ✓

$f(1.5) = +0.25$

$f(1.25) = -0.4375$ ✗

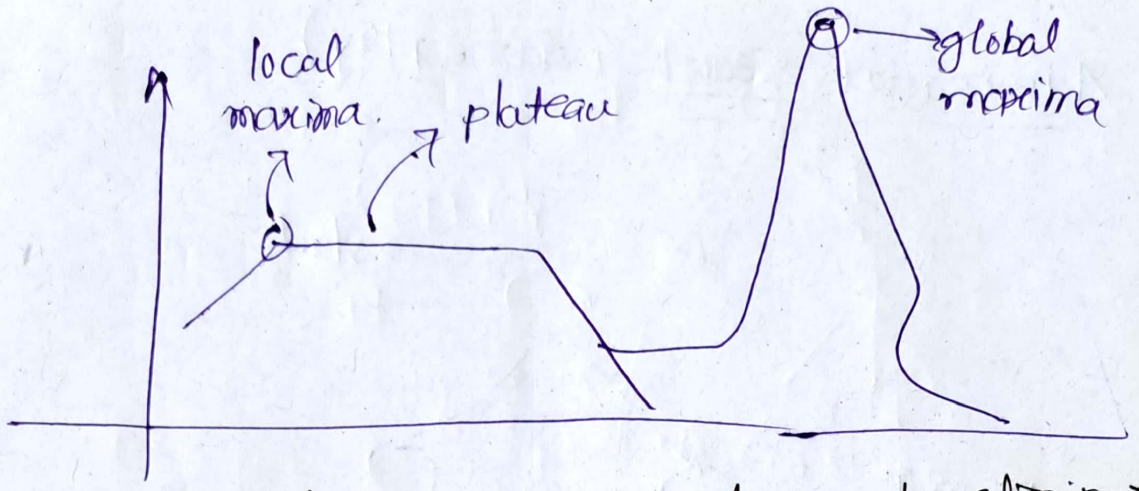
$f(1.375) = -0.109375$] ✓

$f(1.5) = +0.25$

$f(1.375) = -0.109375$] ✓

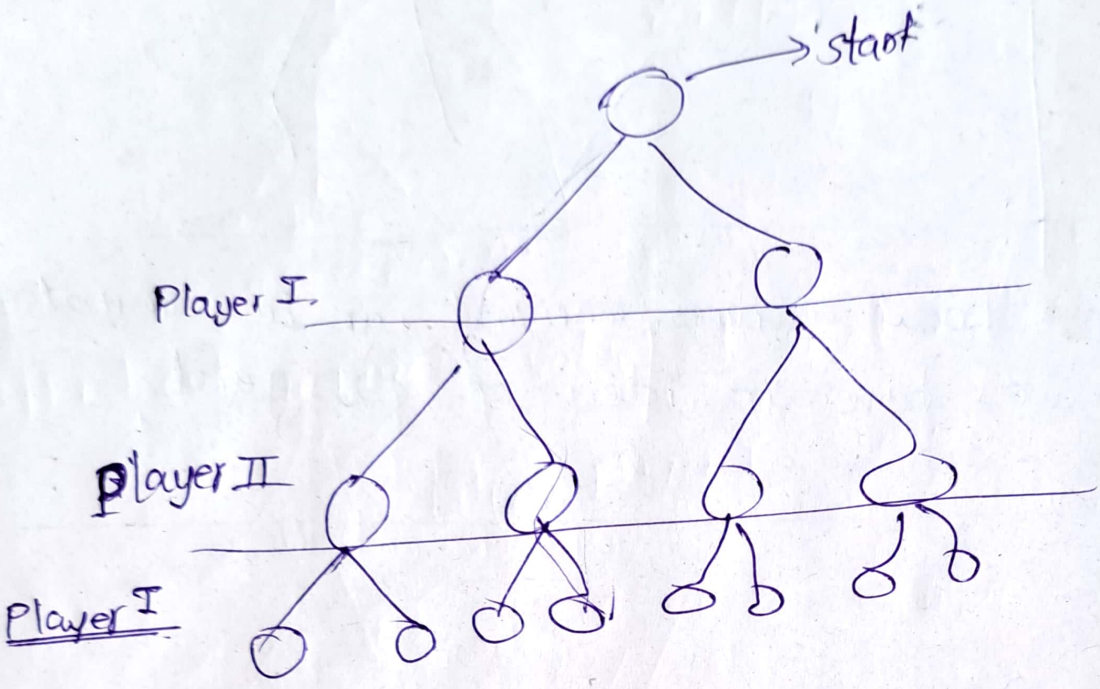
$f(1.4375) = +0.06640625$

$f(1.5) = +0.25$] ✗



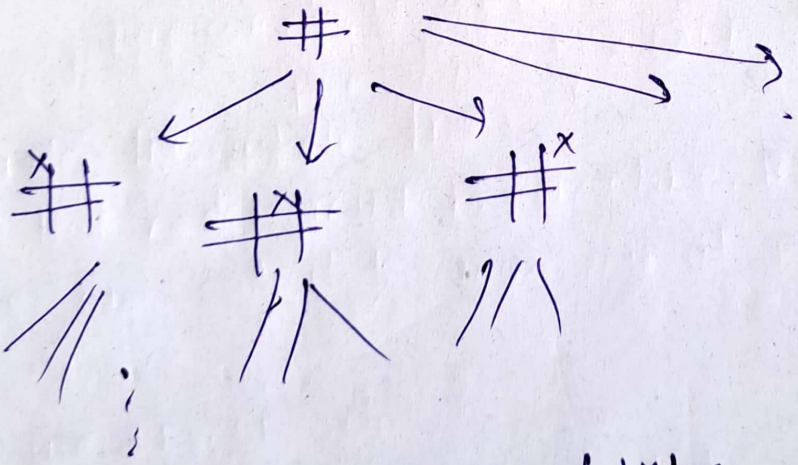
Local maxima, minimal can be eliminated if we have an idea of goal.

Adversarial Search (Game play).



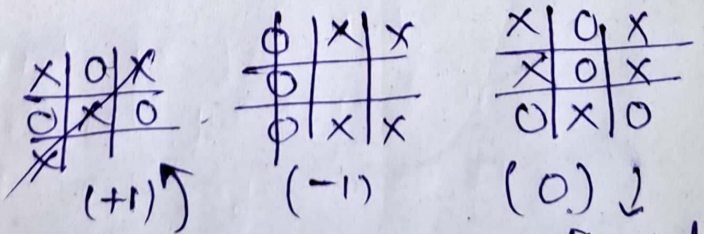
Win / Lose / Draw
1 / 0 / 0.5, 0.5

- We are not finding 1st goal.
- We are finding goal with maximum reward.



Finally 3 cases.

Limited depth :-



max. reward at min no of steps

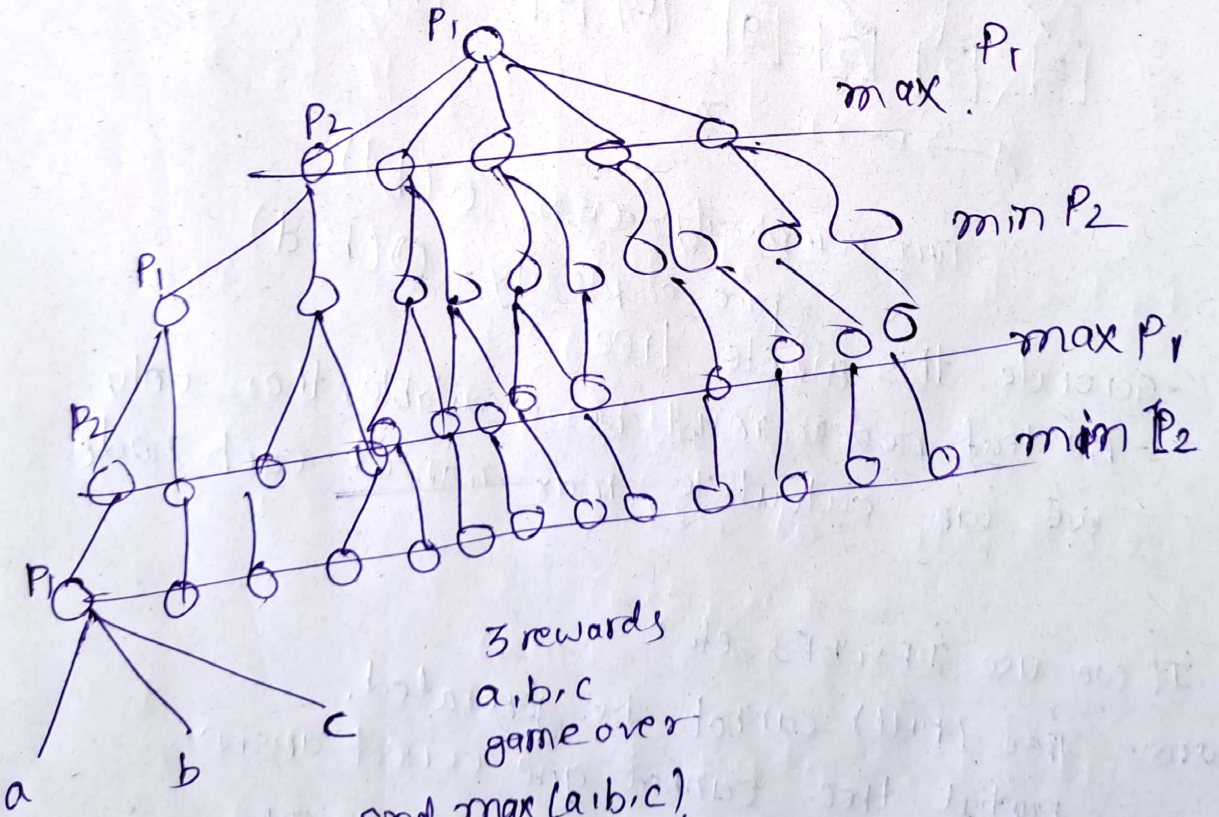
* These game trees will be solved using mini-max algo

zero-sum game tree

	P_1	P_2	sum
win	1	-1	0
Tie	1/2	-1/2	0

some games, may not have same no. of branches (ex- chess, at a point you cannot move anywhere (sometimes you can move in all directions, etc).

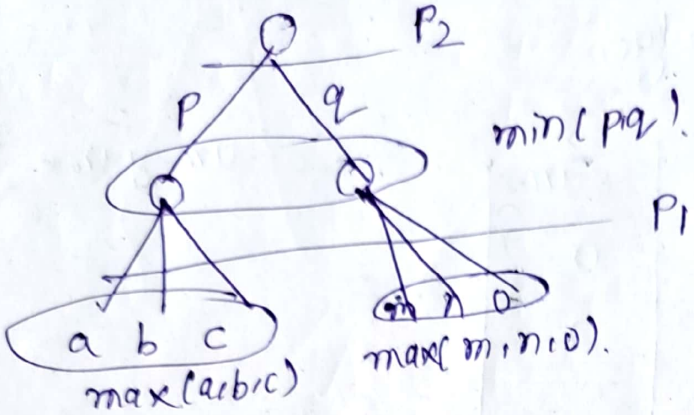
Try to maximize moves of player I.
& minimize moves of player II.



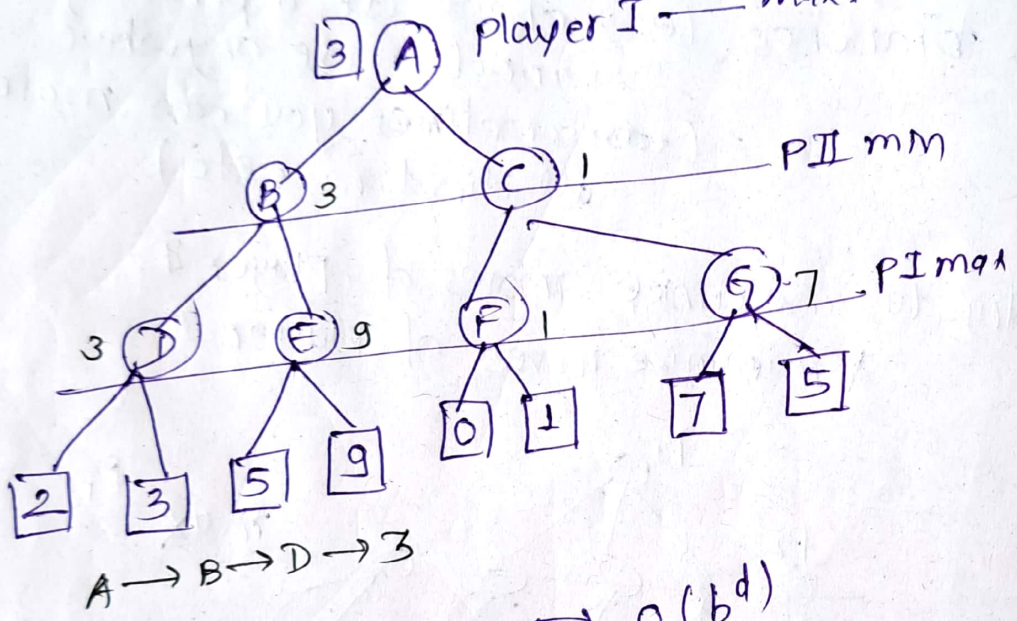
Find max (a, b, c).

If only 1 branch → 1 reward game over

If more paths → max. of the rewards, game over.



a] Find out maximum value at Goal node, 4



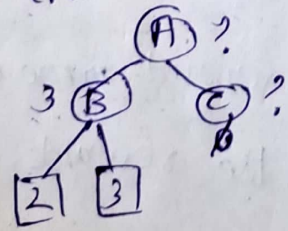
Time complexity:- ~~O(n)~~ $O(b^d)$

Space complexity:- $O(b \cdot d)$

- Generate this whole tree
- All leaf nodes must be available, then only we can calculate max min at each node.

If we use DFS, BFS, etc?

ans:- Tree (full) cannot be generated.
Partial tree cannot give correct answer.
say used DFS.

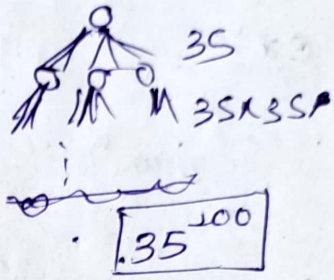


but in games like chess where each player plays 50 moves in average.

∴ (100 moves, player + enemy)

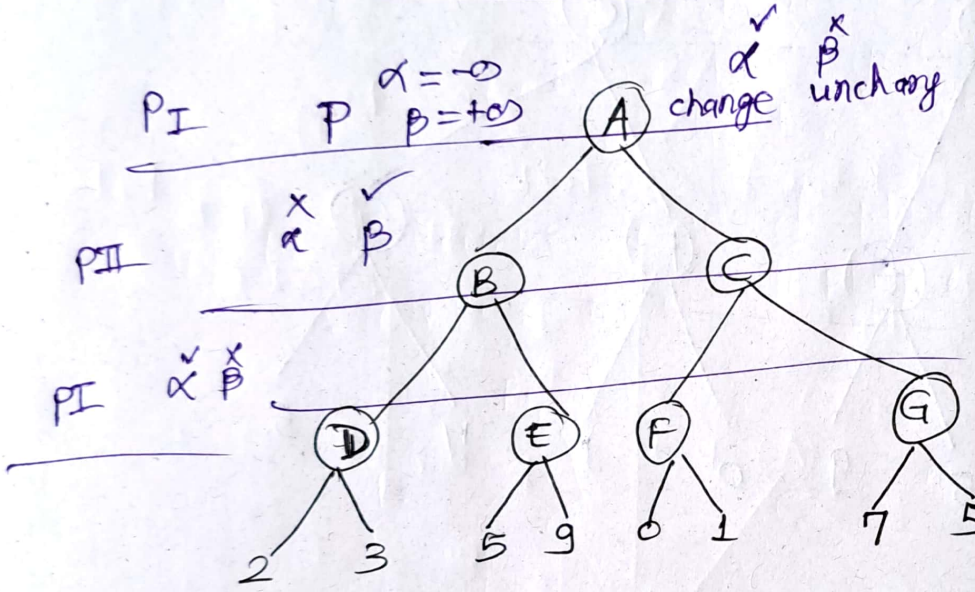
and 35 branches per more (in average)

∴ 35^{100} nodes.



such tree cannot be generated & stored.

We use α-β pruning in such case.



α-β pruning

cut the tree. so that some nodes are not visited

P_1 - α - max

P_2 - β - min

Let initially $\alpha = -\infty$

$\beta = +\infty$

goal

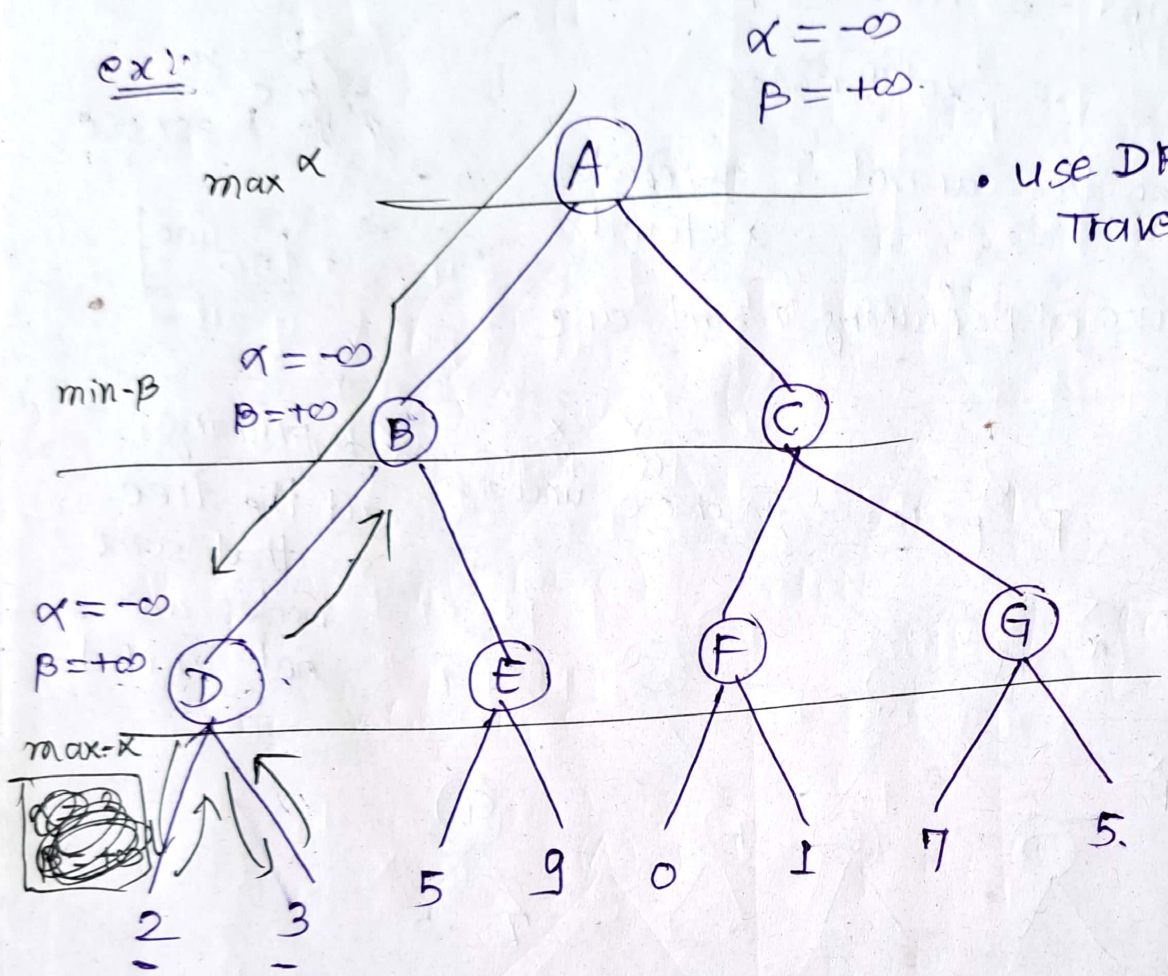
stage 1) all α-β same values will be passed as it is, till Leaf nodes.

stage 2) calculate α, β for each node, while coming back.

If some node follows $\alpha \geq \beta$, prune its child nodes.

node value of a Node = max or min of child reward values.

ex:



First calculation, at point 2

$\alpha = \max(2, -\infty) = 2$

β = unchanged (+∞)

second calc, at point 3

$\alpha = \max(2, 3) = 3$

β = unchanged (+∞)

3rd calc, at point D.

$\alpha = \max(2, 3) = 3$

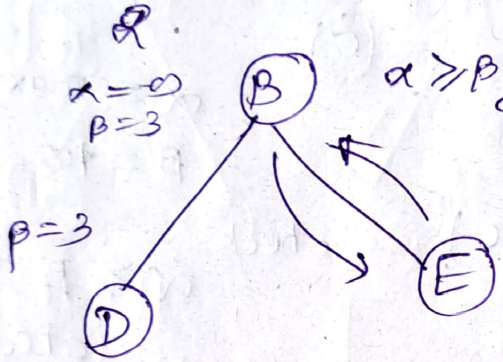
β = unchanged (+∞)

4th calc, min, β, at point B.

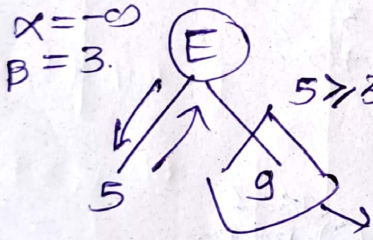
~~unchanged (3)~~

previous values	data from child D:-
$\alpha = -\infty$	$\beta = \min(3, +\infty) = 3$
$\beta = +\infty$	~~unchanged~~ (

Finally $\boxed{\alpha = -\infty, \beta = 3}$



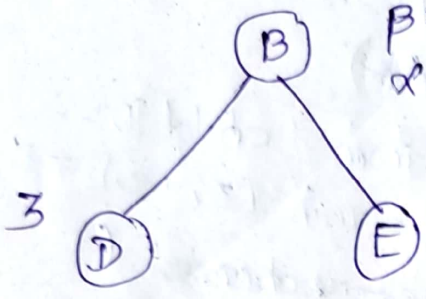
at E, α will change.



$\alpha = \max(-\infty, 5)$
$= 5.$

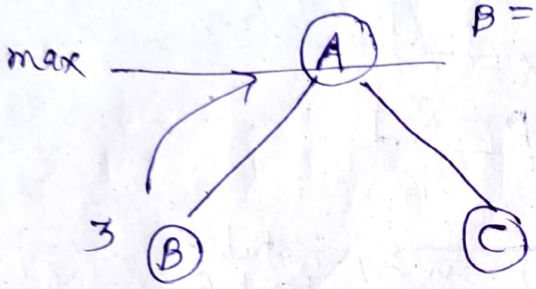
why? min of 3 and (5,9) will be 3 itself
Hence, we cut those branches.

∴ $\boxed{\alpha = 5, \beta = 3}$ (E) ~~root~~ node with 1 reward value



$\beta = \min(5, 3) = 3$.
$\alpha = -\infty$

$\alpha = -\infty$ initially
$\beta = +\infty$



Node value of B = 3

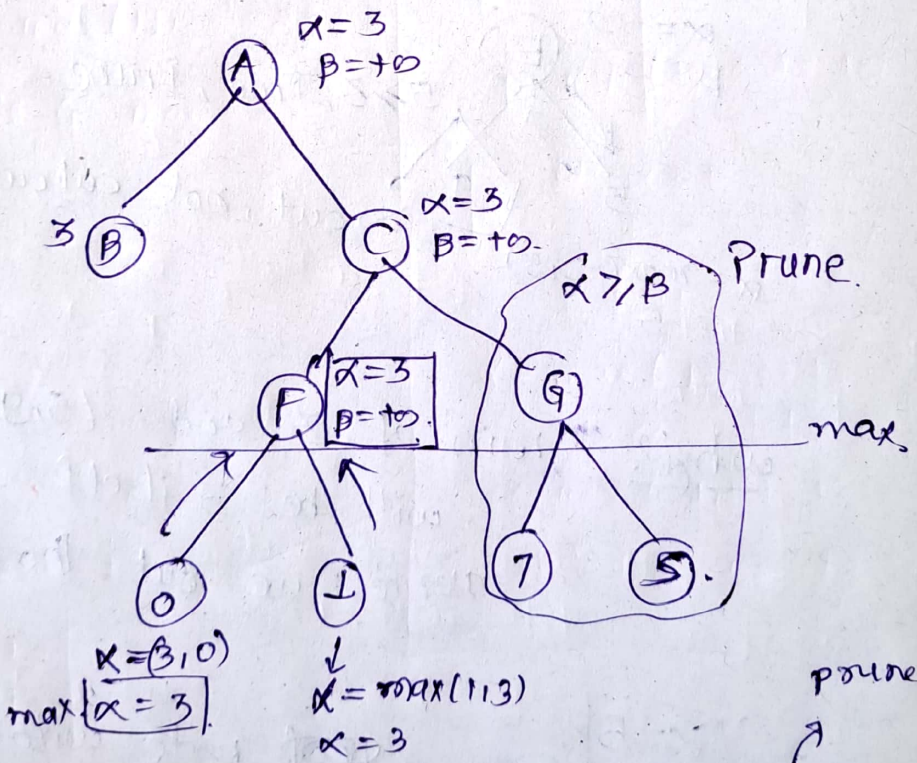
$\therefore \alpha = \max(3, -\infty) = 3$
$\beta = +\infty$

Node value of A = max (B, C).

check $\alpha \geq \beta$? False.

check C.

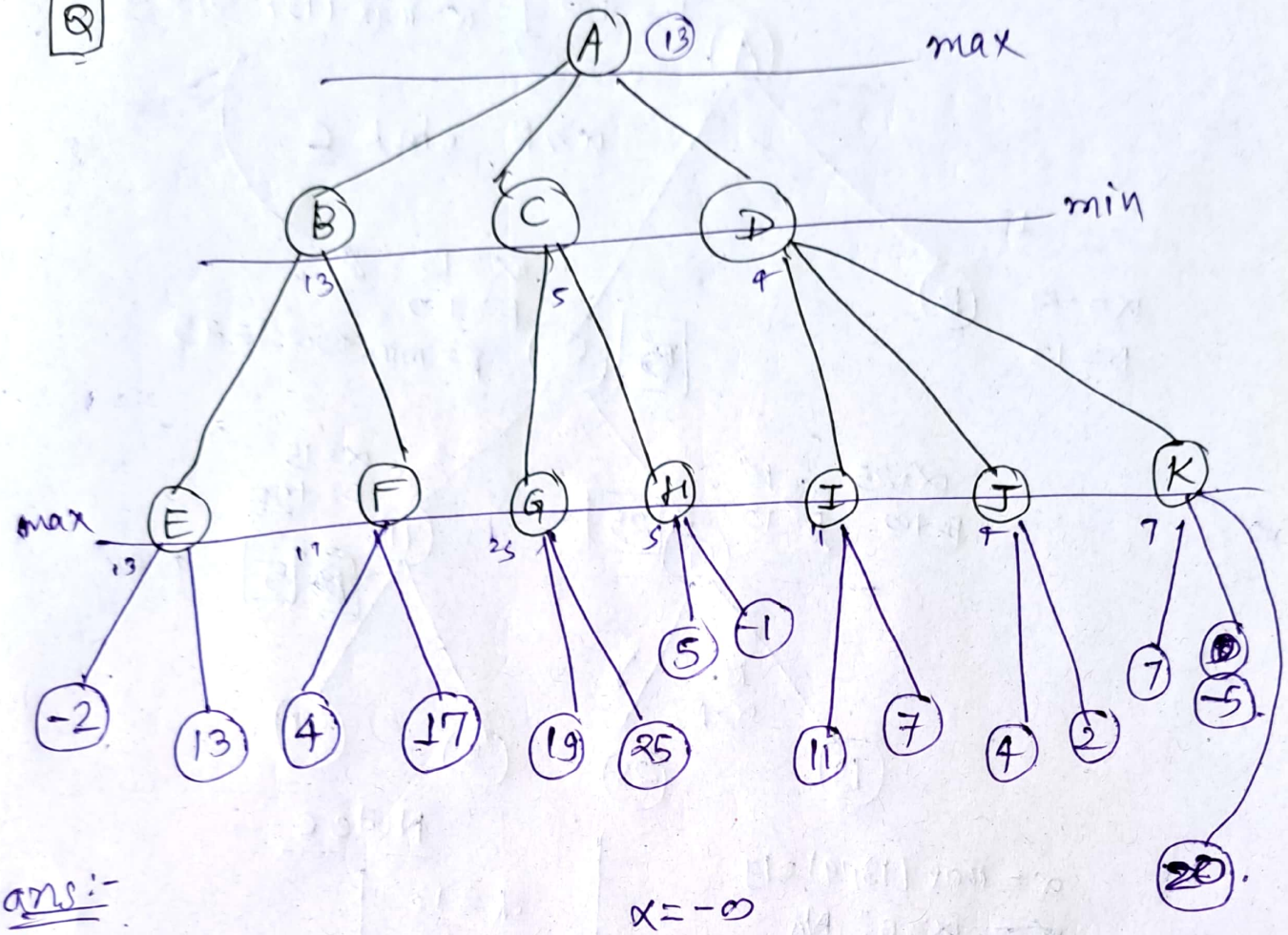
Pass values to C ?



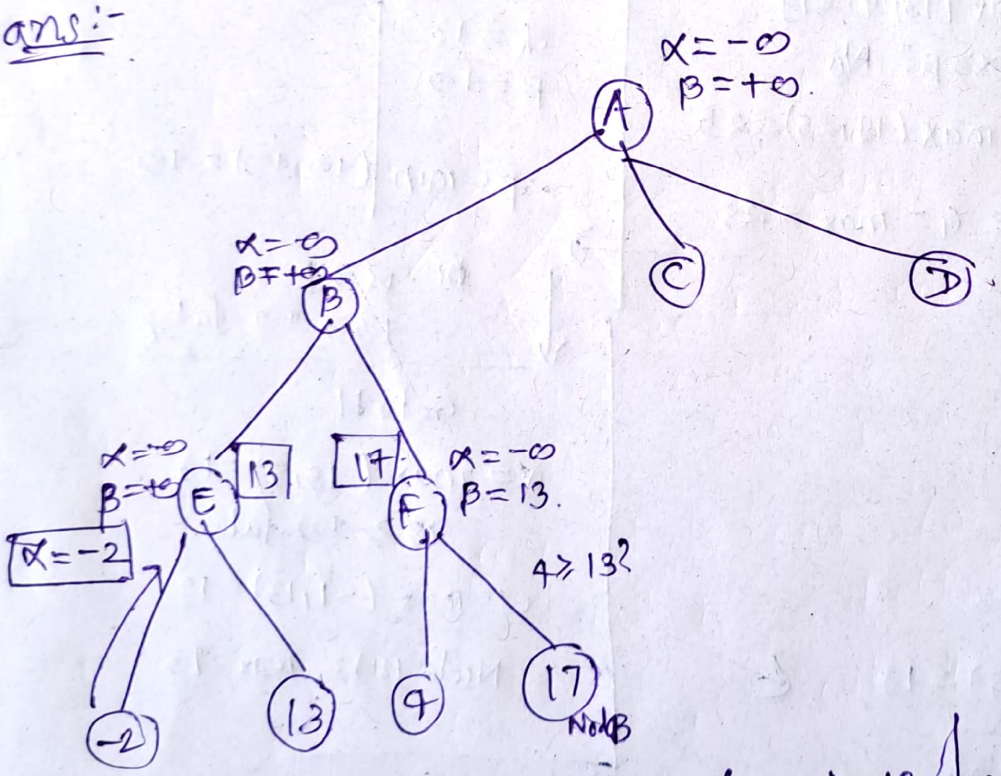
Finally Node value of C = min (node F, G)
= min (3)
= 3

∴ Node value of A = max (B, C) = max (3, 3) = 3.

Q



ans:-



Node B

$\alpha = \max(-\infty, 2) = -2$

$\alpha = \max(-2, 13) = 13$

node(E) = 13

$\beta = \min(13, +\infty) = 13$

$\alpha = -\infty$

check $\alpha \geq \beta$

$-\infty \geq 13$

Node F = 17

$\alpha = \max(-\infty, 4) = 4$

$\alpha = \max(4, 17) = 17$

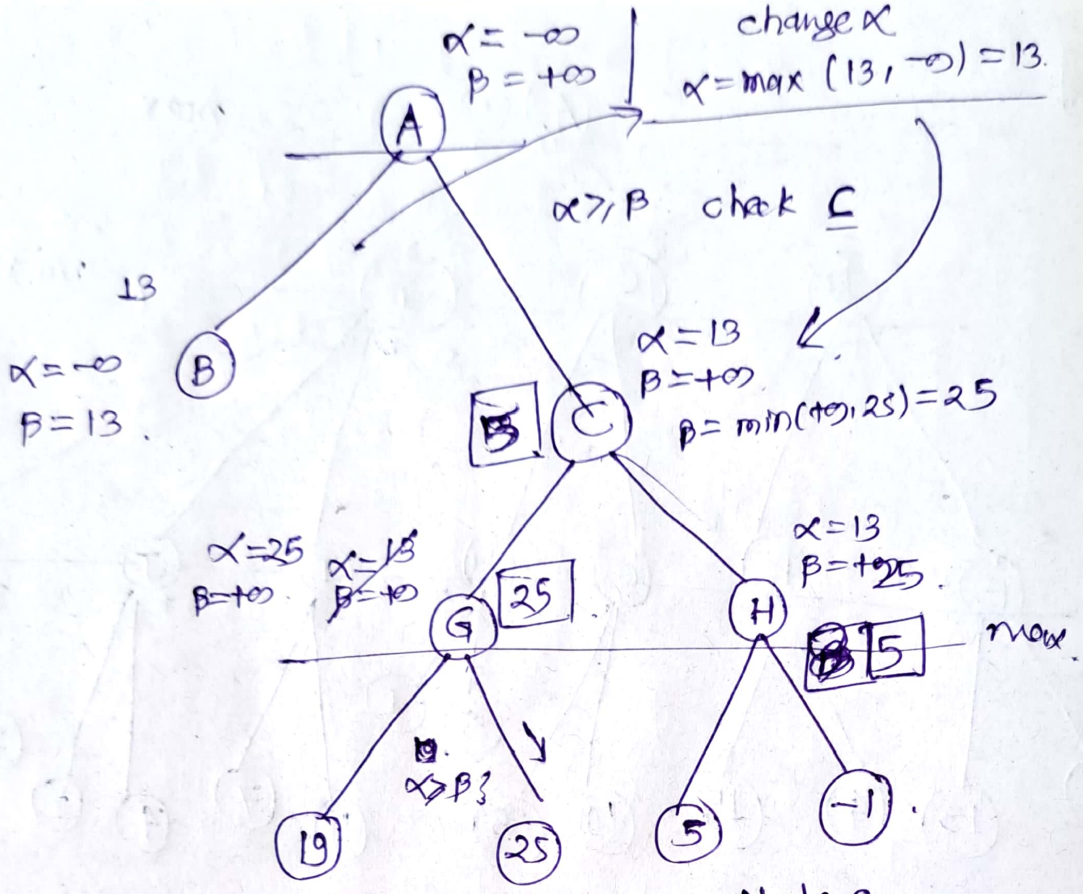
Node B

Node B = min (17, 13)

= 13

$\alpha = -\infty$

$\beta = 13$



Node C

$\alpha = \max(13, 19) = 19$

$\alpha \geq \beta$? No

$\alpha = \max(19, 25) = 25$.

Node G = max = 25.

Node C

$\alpha = 13$

$\beta = +\infty$.

$\therefore \beta = \min(+\infty, +\infty) = +\infty$.

$\alpha \geq \beta$

$13 \geq +\infty$ false

Goto H

$\alpha = \max(13, 5) = 13$.

$13 \geq +\infty$ false

$\alpha = \max(-1, 13) = 13$

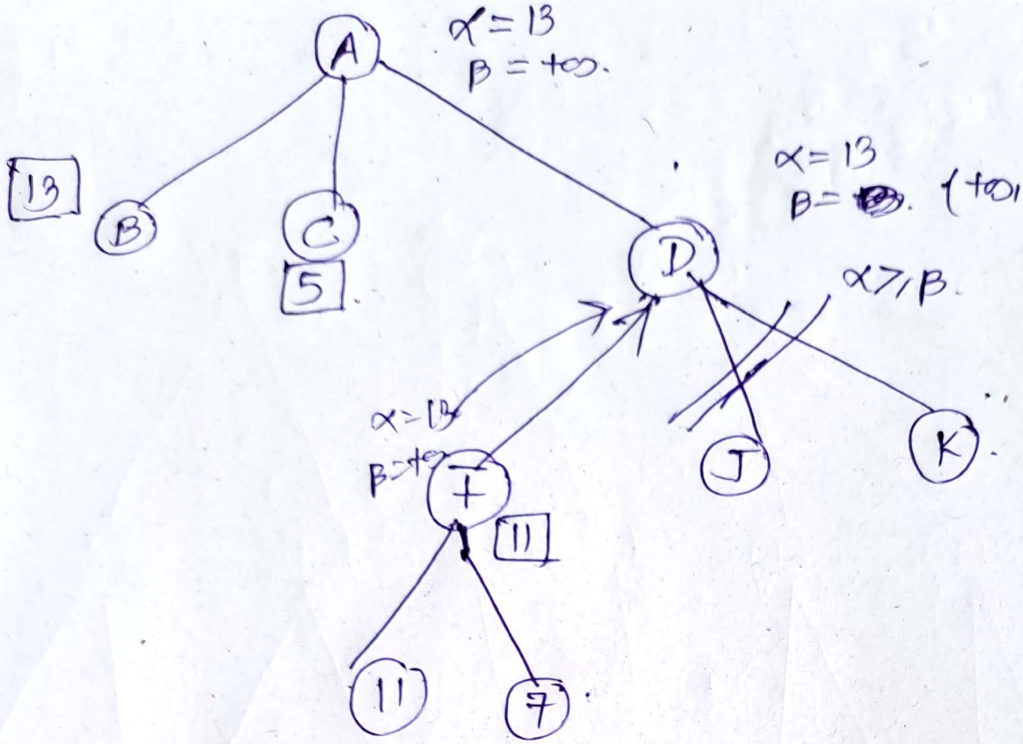
Node (H) = max 13.

Node C

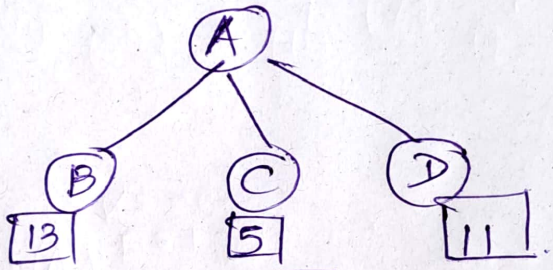
β, min

$\beta = \min(13, 25) = 13$.

$\therefore \boxed{\alpha = 13, \beta = 13}$



∴ Finally



α = max (13, 5, 11) = 13.

Last class before midsem.

All topics discussed till now will be asked.

common topics of all section will be asked.

Searching Methods ✓

Beam search ✗ (Trousers, Tracksuit, chappal)

---

object :- an entity having certain attributes

predicates :- property

statement :- is a sentence with a Truth value. (T/F)

. don't involve ?, ! (question, exclamation).

ex:- Today is Thursday. True

This board is white. True & ambiguous

↳ context dependent.

Proposition :- A sentence with either True or false value.

p: Sky is blue. propositions are used by robots to infer logic.

ex:- Robita lab ccd. for robotics

p: is a student.

p(x) :- x is student.

p(Bhupesh) :- Bhupesh is a student.

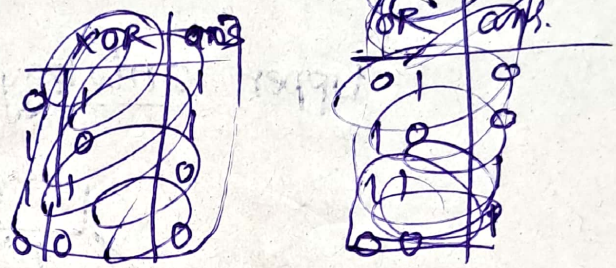
operators	connectors	
NOT	¬	
AND	∧	conjunction
OR	∨	disjunction
NAND	↑	¬∧
NOR	↓	¬∨

P: Today is Thursday.
¬P :- Today is not Thursday

XOR ▽ exclusive OR.

inclusive OR?

ans:- OR itself.



property :- P(x)
predicate :- P

R(z) = P(x) ∧ Q(y).

connective

if then →

(iff) if & only if ↔

p	q	p → q	p ↔ q
F	F	T	T
F	T	T	F
T	F	F	F
T	T	T	T

universal Truth
∀x P(x)

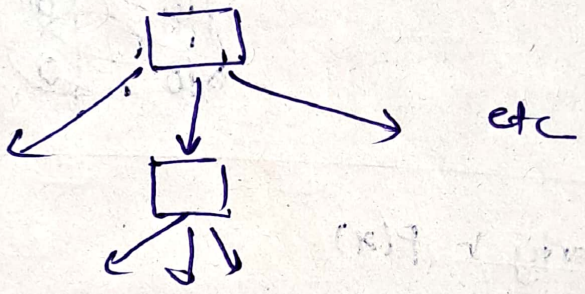
Existential
∃x Q(x)

Toy world → Games (Board)

Real world → Agents

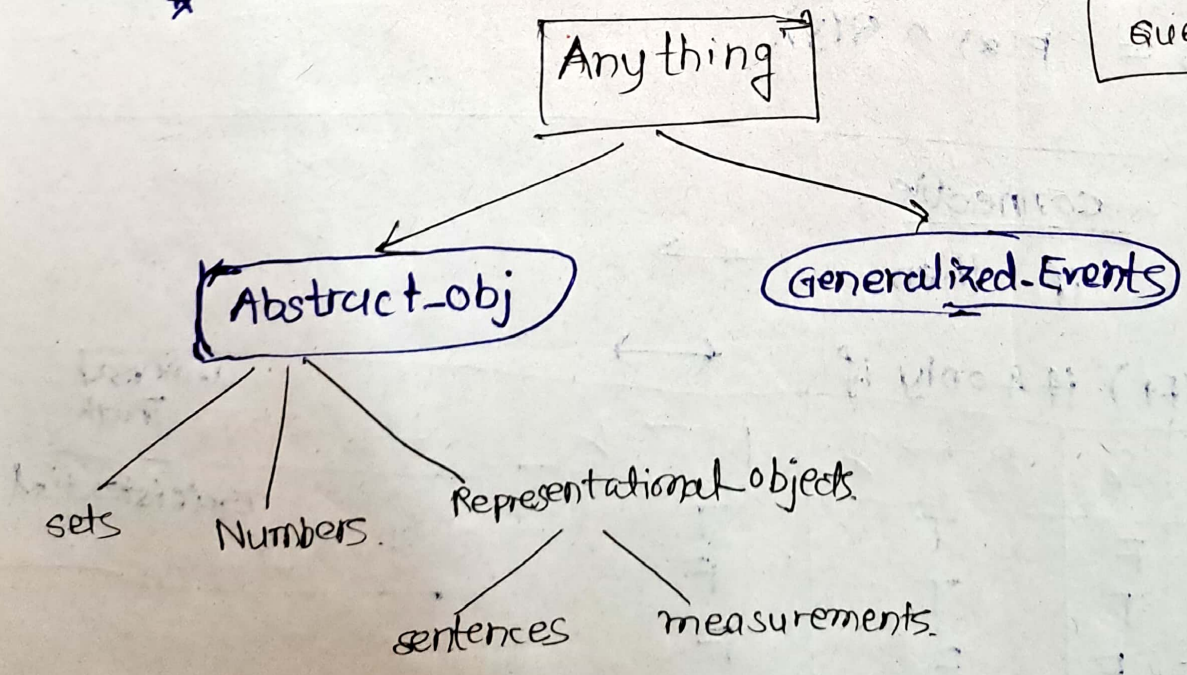
Ontological diagram
- events
- time
- physical objects.
- belief

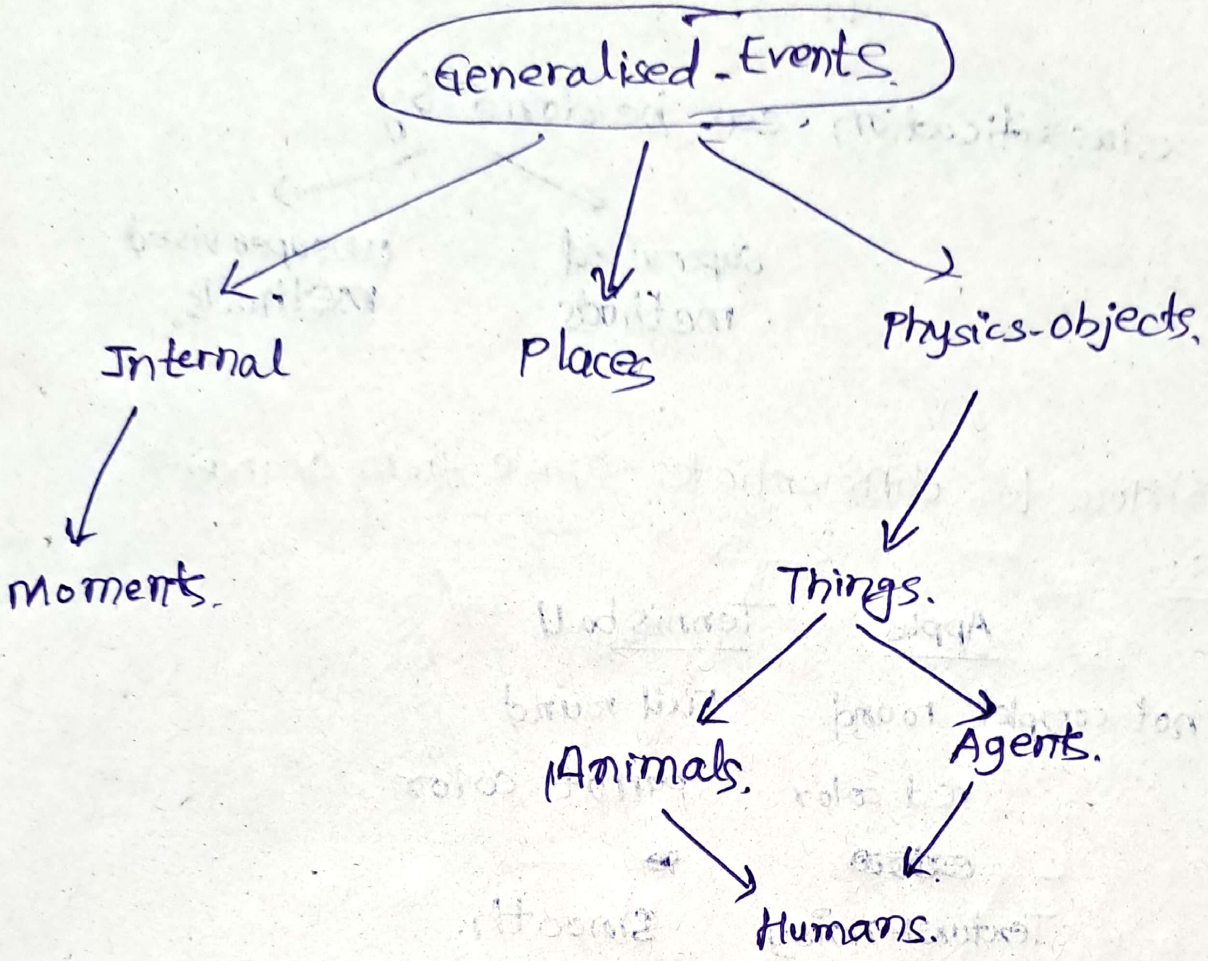
Upper ontology : convention



* everything in real world cannot be represented.
*

#Exam question





Object based :- represented as graphs.

predicate based :-

Taxonomy :- classification.

classification can be done by

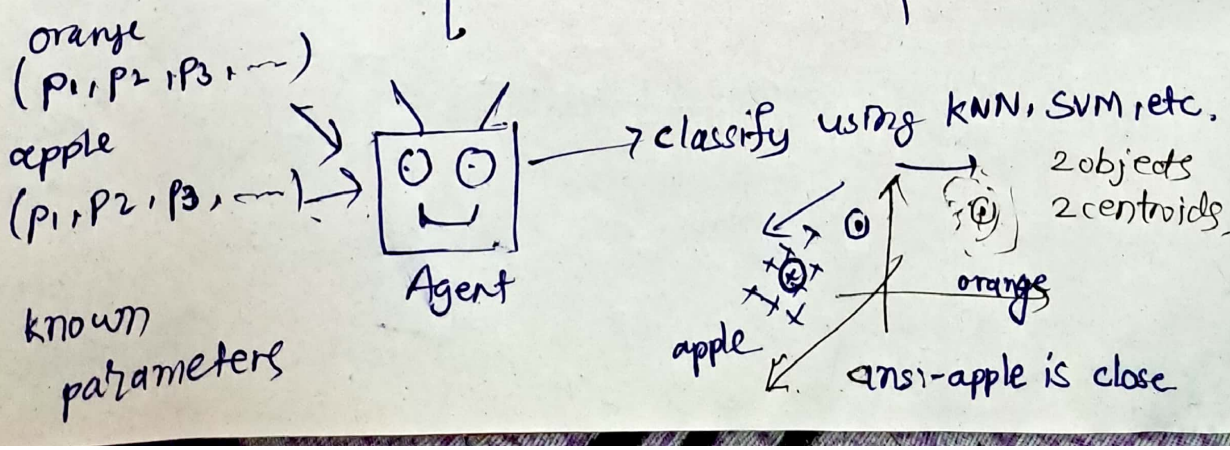
- supervised methods
- unsupervised methods

Q) How to differentiate apple from oranges?

ans :-

Apple	Tennis ball
not complete round	Full round
red color	parrot color.
Texture :- noisy.	smooth.

Apple	orange	
red	orange	P1
noisy yellow-red	same color all over	P2
apple is smooth.	orange is rough	P3



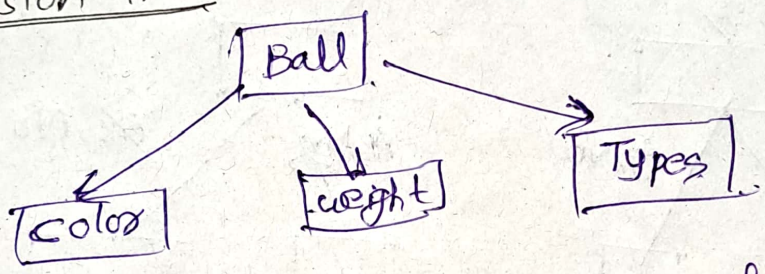
Knowledge in form of text.

~~Text~~ → predicate & operators ⟹ Logical Inference

Gives ontological graph, useful for classification.

Knowledge Representation

Decision Tree



- we can have different classes in form of Leaves.

Decision Tree Algorithm

i/p — D (training set), C

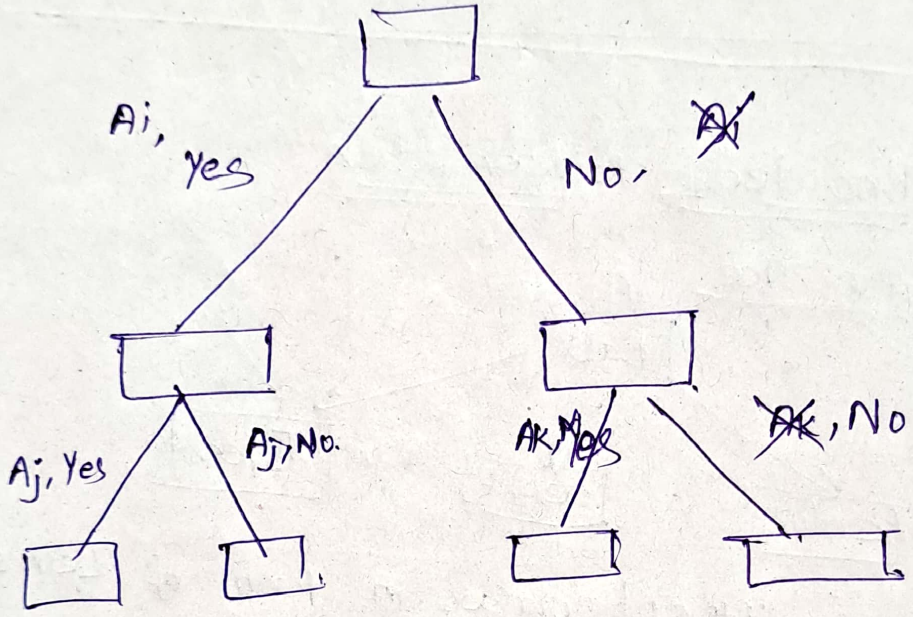
o/p — T (decision tree)

1) If all types $\in C_j$, Add a leaf labelled C_j ~~return~~
   return
2) select A_i (attribute)
3) Partition $D = [D_1, D_2, \ldots D_x]$ based on A_i
4) $\forall D_K \in D$
   create a node & add an edge
   $D_K - D$, label: A_i

5. $\forall D_k \in D$

$D^{\dot{T}}(D_k)$

Stop



Complex Decision Making

constraint satisfying problem

based on x, calculate value of $f(x)$.

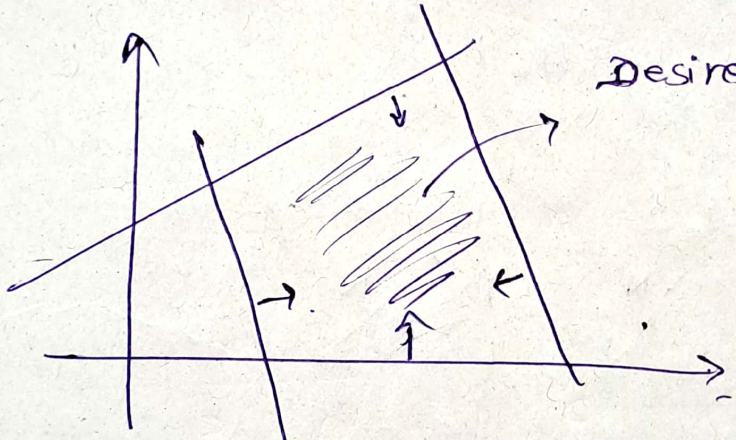
~~such that~~

ex:- $f(x,y) =$

constraint s.t $x \geq ?$ $x < y$

$y \geq ?$ $\leq y \leq$

In class 10th, draw all lines graphically



* fails in 3 variables.

For 3 variables

- Keep 2 variable constant (say y & z)
- change value of x (as per constraints on x)

  blind search

  $x > 0$:- x: 1, 3, 7, 15, 30, ...

  move ~~see~~ only if new x is better.
- If x optimized, make it constant.
- Do same for y and z.

Exam question

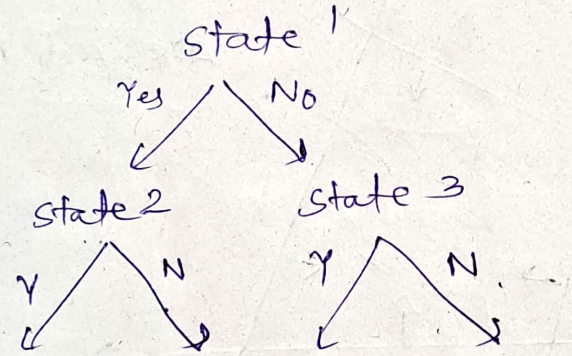
1. ① constraint satisfying problem.
   ② complex decision making.

What we will study ?

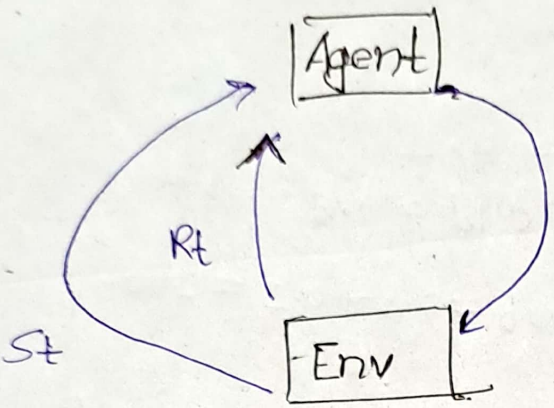
- Markov model
- Probabilistic Reasoning.

H.W. Decision Tree, Monday.

Decision Tree :- Binary Tree.



States included in Agent working diagram :-



$S_0 \longrightarrow S_1$ ✓

$R_t \Rightarrow$ Reward

$R^+ \rightarrow$ reward

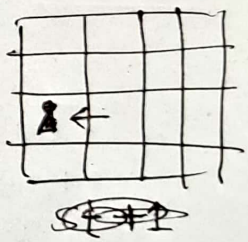
$R^- \rightarrow$ punish

Markov Model.

- Next state will either be deterministic or non-deterministic.

chessboard

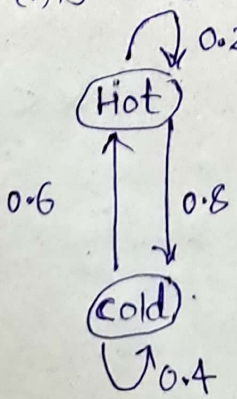
S_t



S_{t+1}

- It can be guessed from probability of that event.

ex:- consider this weather situation.



S_t = Hot (say)

S_{t+1} = cold.

S_{t+2} = Hot.

Cold remaining cold probability = 0.4

Cold not remaining cold = 0.6

Total probability = 0.4 + 0.6 = 1

Similarly.

Hot remaining hot = 0.2

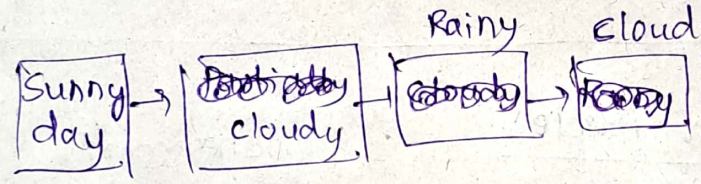
Hot changing to cold = 0.8.

Total prob = 0.8 + 0.2 = 1.0

If probability of next state is high, move to S_{t+1}.

If not, it remains in same state.

Weather Forecasting



- This will be represented in Transition matrix.
- Row sum will be 1.

$\Sigma = 1$

$$\begin{bmatrix} 0.8 & 0.2 \\ 0.6 & 0.4 \end{bmatrix}$$

row x column

r x c

Lets write matrix Neatly with notation

	H	C
H	0.2	0.8
C	0.6	0.4

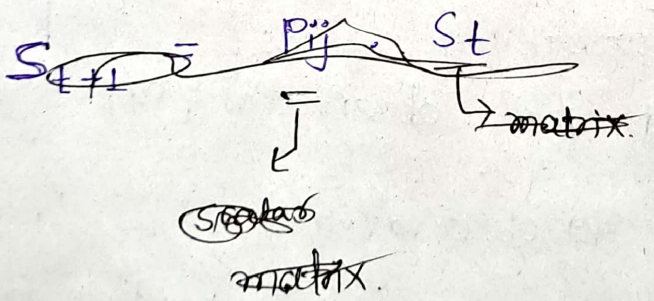
S-sunny
C-cloudy
R-Rainy

Mapping this to real world

	S	C	R	C
S	0.2	—	—	—
C	—	0.1	0.3	—
R	—	—	0.3	as
C	—	—	0.1	0.4

etc.

Next State is predicted like this:-



$$S_{t+1} = S_t \cdot P$$

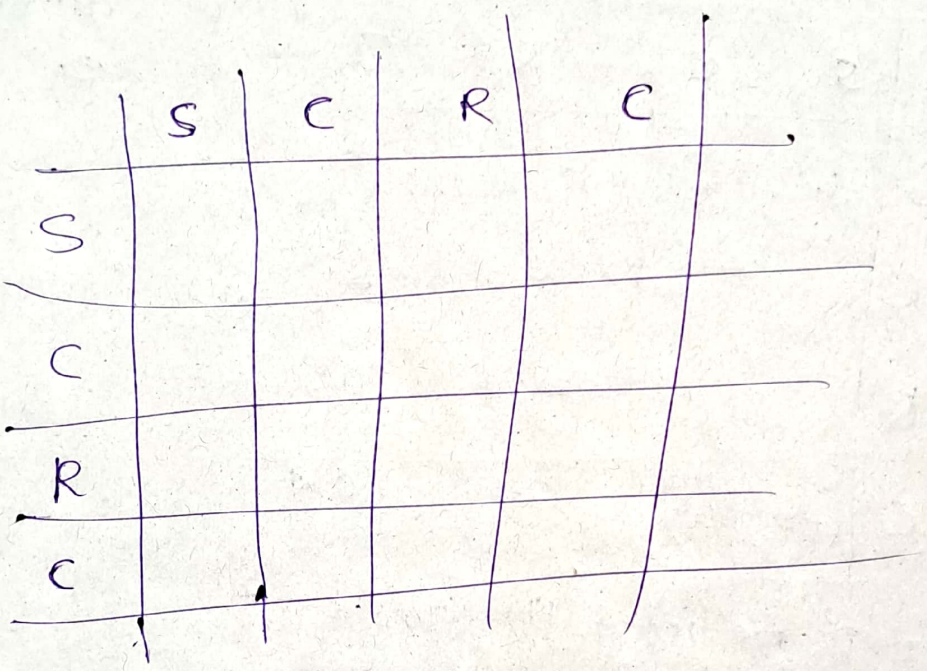
S_{t+1} → row vector.

P → square matrix.

Take example of student getting job or not

$$\begin{bmatrix} 0.5 & 0 & 0.5 & 0 \end{bmatrix}$$

Job | No Job | Startup | Higher studies.



How do we get probability values?

ans:- Guessed from observation & experiments

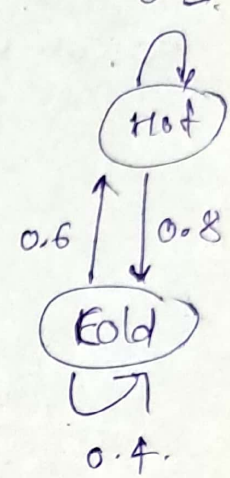
Factors for observation:-

ex:- Each decision affected by multiple factors.

Factors → wind direction
→ study atmosphere.
→ western disturbance
→ ...etc. → pollution.

whether next day will be sunny or not.

No of values in state vector = no. of states.



$$S_{t+1} = S_t \cdot P_{ij}$$

$$= \begin{matrix} & H & C \\ S_t & \end{matrix} \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{matrix} H \\ C \end{matrix} \begin{bmatrix} 0.2 & 0.8 \\ 0.6 & 0.4 \end{bmatrix}$$

$$S_{t+1} = \begin{matrix} H & C \end{matrix} \begin{bmatrix} 0.2 & 0.8 \end{bmatrix}$$

#will be asked in exam, no calculator allowed.

$$S_{t+2} = S_{t+1} \cdot P_{ij}$$

$$= \begin{bmatrix} 0.2 & 0.8 \end{bmatrix} \begin{bmatrix} 0.2 & 0.8 \\ 0.6 & 0.4 \end{bmatrix}$$

$$= \begin{bmatrix} 0.52 & 0.48 \end{bmatrix}.$$

$$S_{t+3} = \begin{bmatrix} 0.52 & 0.48 \end{bmatrix} \begin{bmatrix} 0.2 & 0.8 \\ 0.6 & 0.4 \end{bmatrix}$$

$$= \begin{bmatrix} 0.392 & 0.608 \end{bmatrix}$$

$$S_{t+4} = \begin{bmatrix} 0.392 & 0.608 \end{bmatrix} \begin{bmatrix} 0.2 & 0.8 \\ 0.6 & 0.4 \end{bmatrix} = \begin{bmatrix} 0.4432 & 0.5568 \end{bmatrix}$$

$$\sim \begin{bmatrix} 0.4 & 0.6 \end{bmatrix}$$

saturated.

We can calculate continuously,
it gets saturated after a limit

Here $[0.4, 0.6]$

$$S_1 = S_0 \cdot P$$

$$S_2 = S_1 P = (S_0 P) P = S_0 P^2$$

$$S_3 = S_2 P = S_0 P^3$$

$$S_{t+1} = S_0 \cdot P^{(t+1)}$$

Hence $S_{t+1} = S_t$

$$\begin{bmatrix} 0.4 & 0.6 \end{bmatrix}$$

H C

Being cold has higher probability
we cannot guess further.

Means, calculations done in march, cannot be applied to november-december.

If states probability matrix should not be kept constant, it has to be modified.

H.W Lab exercise.

* Markov model is useful for probabilistic planning & reinforcement Learning.

* Markov decision process is stoachastic decision process.

MDP

set of states $S = [S_0, S_1, \ldots S_t]$

set of possible actions $A = [A_0, A_1, \ldots]$

set of reward values $R = [R_0, R_1, \ldots -]$

description of each action T repeesented as P_{ij}.

Transition probabilities.

Markov property :- Next state will be depended on present state, not in the past.

S_i depends on S_{i-1}, but not on S_{i-2}, S_{i-3}, etc.

advantage :- calculation based on 1 state

if 5th degree markov :- S_i, depends on its 5 previous states

i.e. calculation based on 5 states

Actions → Deterministic : $S \times A \rightarrow S$.

Actions → stochastic : $S \times A \rightarrow P(S)$

- Deterministic will be followed for writing Agent programs.
- In Real life → stochastic is followed.

All of this is hypothesis.

How good is policy? – will be shared on Google classroom

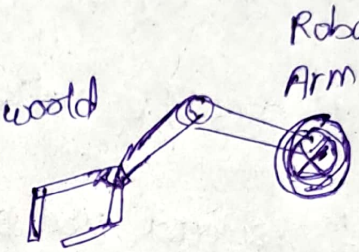
Policy? is a mapping from $S \rightarrow A$

$$\pi = S \rightarrow A$$

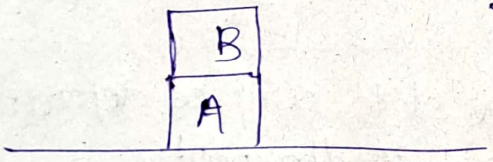
Planning

Knowledge representation (predicates)
Actions.

Block world (toy)



initial state.



final state

- Move these blocks
- write down actions.

Agent → state

6] Write situation in form of predicates

ON (A, B)
CLEAR (A)
HOLDING (A)

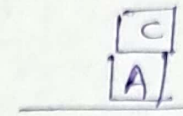
ans:-

ONTABLE (A) ∧ ONTABLE (B)

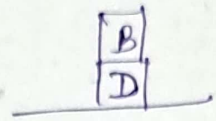
---

will be asked in exam

ON (C, A) C is ON A

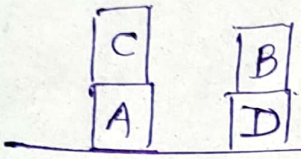


ON (B, D) B is ON D



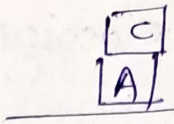
ON (C, A) ∧ ON (B, D) ∧ ONTABLE (A) ∧ ONTABLE (D)

goal stack.

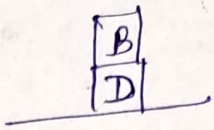


goal stack has
- each actions
- so some statements can be repeated.

ON(C,A) C is ON A



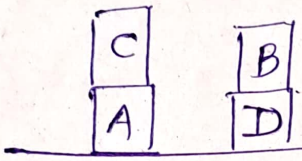
ON(B,D) B is ON D



ONTABLE(A) ∧ ONTABLE(D)

ON(C,A) ∧ ON(B,D) ∧

goal stack.

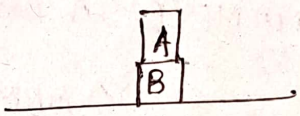


goal stack has
- each actions
- so some statements can be repeated.

---

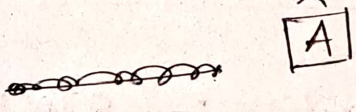
ex:-

ON(A,B)



ONTABLE(A)

CLEAR(A)



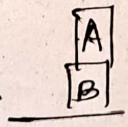
no block on top of A.

HOLDING(A) Arm holds A

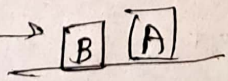
ARM EMPTY Arm is empty

Actions

Stack(A,B)



Unstack(A,B)



Pickup(A)

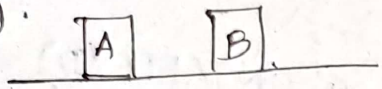
Putdown(A). → To place block on Table

:- Planning (classical) :-

State

• ONTABLE(A) ∧ ONTABLE(B)
∧ CLER(A) ∧ CLEAR(B)
∧ ARMEMPTY

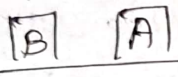
(1)



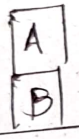
initial.

||

identical state



(2)



ONTABLE(B) ∧ ON(A,B) ∧ CLEAR(A)
∧ ARMEMPY

final.

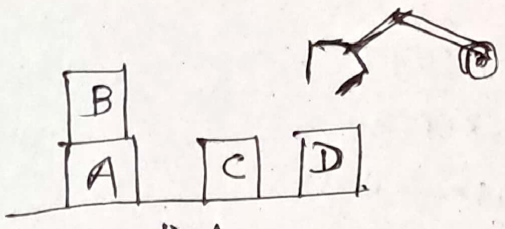
* We need action :- STACK(A,B) ✓
this action requires some pre-condition
like ARMEMPTY, etc.

ARMEMPTY
PICKUP(A)
HOLDING(A)
~~PUTD~~ STACK(A,B)
ARMEMPTY

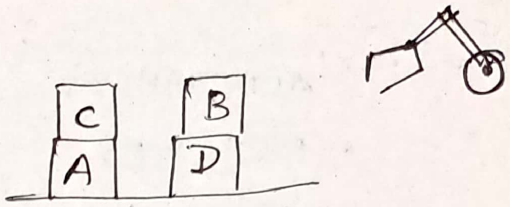
→ combination of action & conditions.

→ represent action by dot on side.

6]



initial



goal.

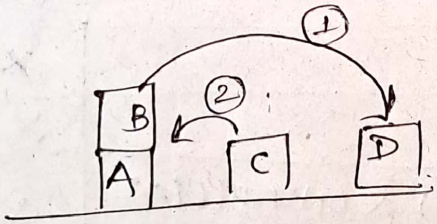
*will come in exam.

state:- ONTABLE(A) ∧
ONTABLE(C) ∧
ONTABLE(D) ∧
ON(B,A) ∧
CLEAR(B) ∧
CLEAR(C) ∧
CLEAR(D) ∧
ARMEMPTY

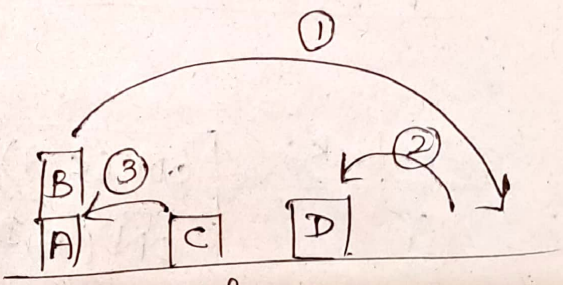
state:- ONTABLE(A) ∧
ONTABLE(D) ∧
ON(C,A) ∧
ON(B,D) ∧
CLEAR(C) ∧
CLEAR(B) ∧
ARMEMPTY

Just extra step for safety.

Possibilities



way 1



way 2

answer UNSTACK(B,A).

precond:- CLEAR(A) ✗

STACK(C,A)

ARMEMPTY ✓
PICKUP(B).
CLEAR(D) ✓
STACK(B,D)
ARMEMPTY ✓
PICKUP(C).
CLEAR(A).
STACK(C,A)

Way 2

ARMEMPTY ✓
- PICKUP(B)
- PUTDOWN(B)

CLEAR(D) ✓
- STACK(B,D)

CLEAR(A) ✓
- STACK(C,A)

or

ARMEMPTY
- PICKUP(B)
- PUTDOWN(B)

ARMEMPTY ✓

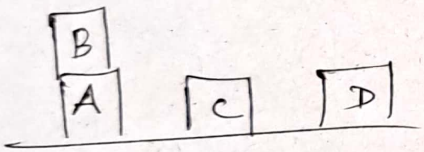
CLER(A) ✓

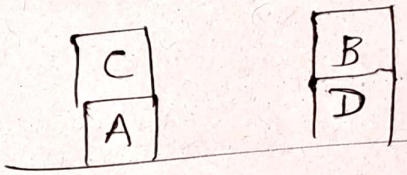
~~STACK(C,A)~~. CLEAR(C) ✓
- STACK(C,A)

CLER(D) ✓

CLER(B) ✓
- STACK(B,D)

This is implemented by Goal ~~stack~~ Stack planning :-





Goal stack :-

	Truth	pre-condition
① ON(C,A)	X	=
ON(B,D)	X	=
ON(C,A) ∧ ON(B,D) ∧ ONTABLE(A) ∧ ONTABLE(D) ∧ ARMEMPTY.	X	=

Now, check whether each one is True or not?

To achieve the subgoals, preconditions will be put, in the side column.

Goal stack (2).

ON(B,D)
ON(C,A)
ON(C,A) ∧ ON(B,D) ∧ ONTABLE(A) ∧ ARMEMPTY ∧ ONTABLE(D)

You can choose any Goal stack, both solutions are valid. (goal stack 1 & goal stack 2)

Lets choose Goal stack 1.:-

ON(C,A) is not true.
Hence, we have to take these actions:-

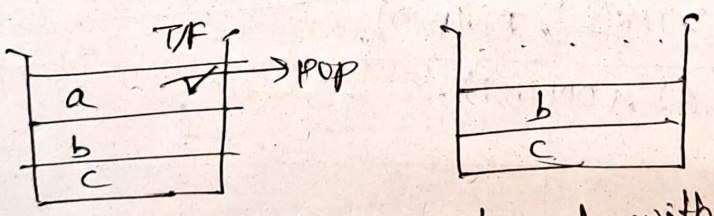
STACK(C,A)

---

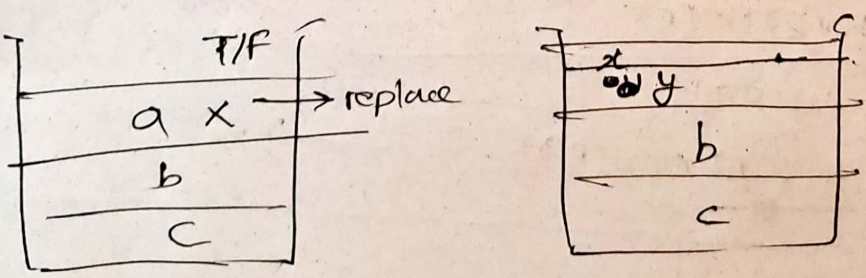
ndition

The goal stack changes

- If something is True, it is popped out of stack. & next actions/goals are evaluated.

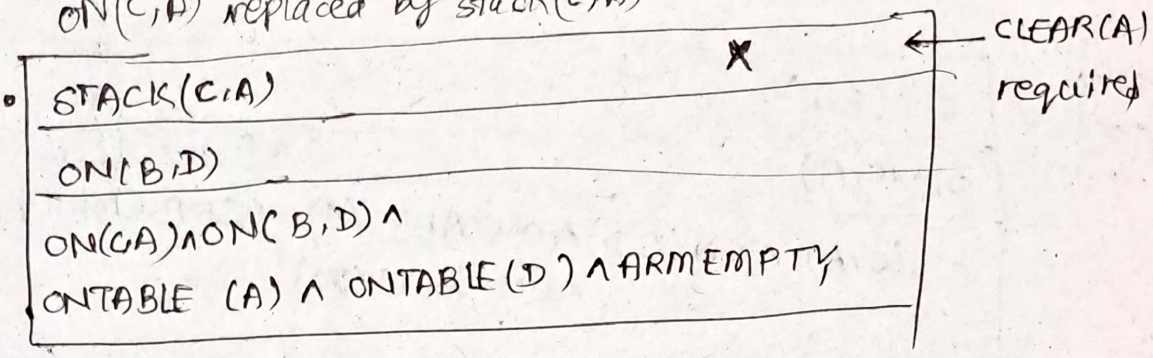


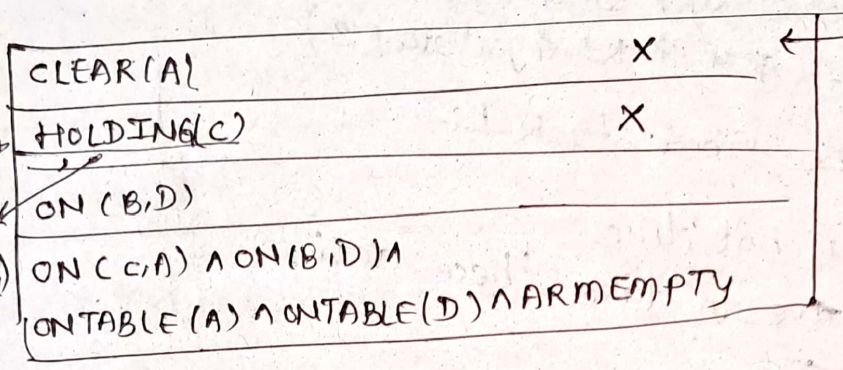
- If goal is false, it is replaced with other actions & pre-conditions

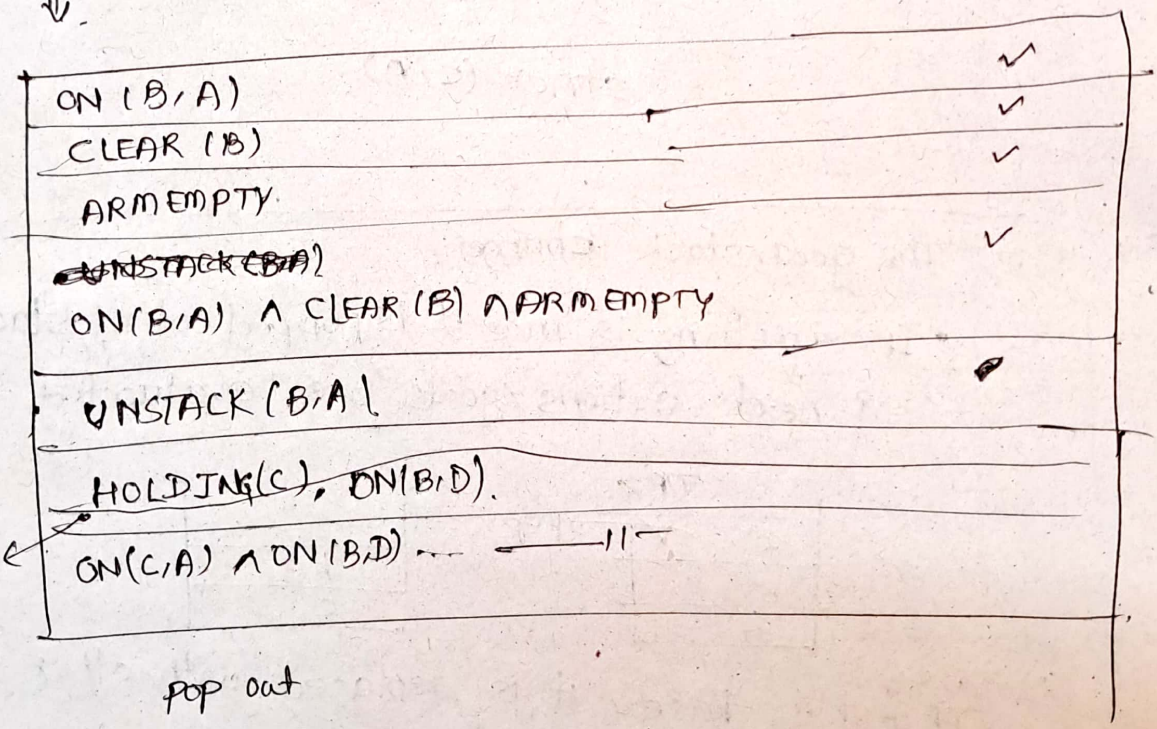


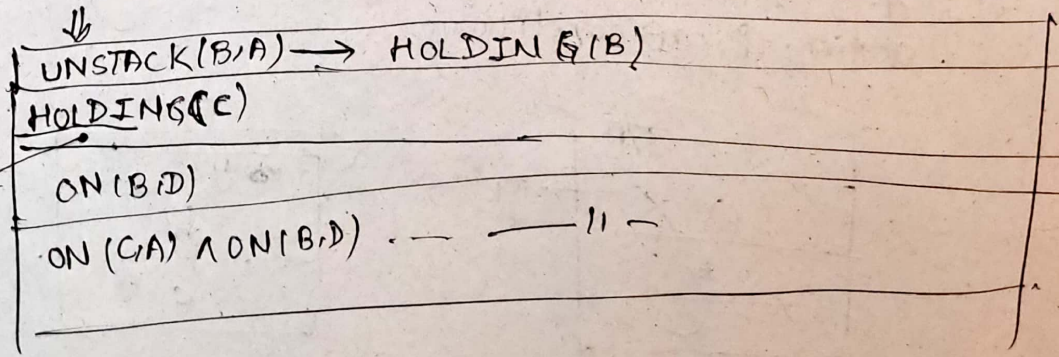
a needs x,y.

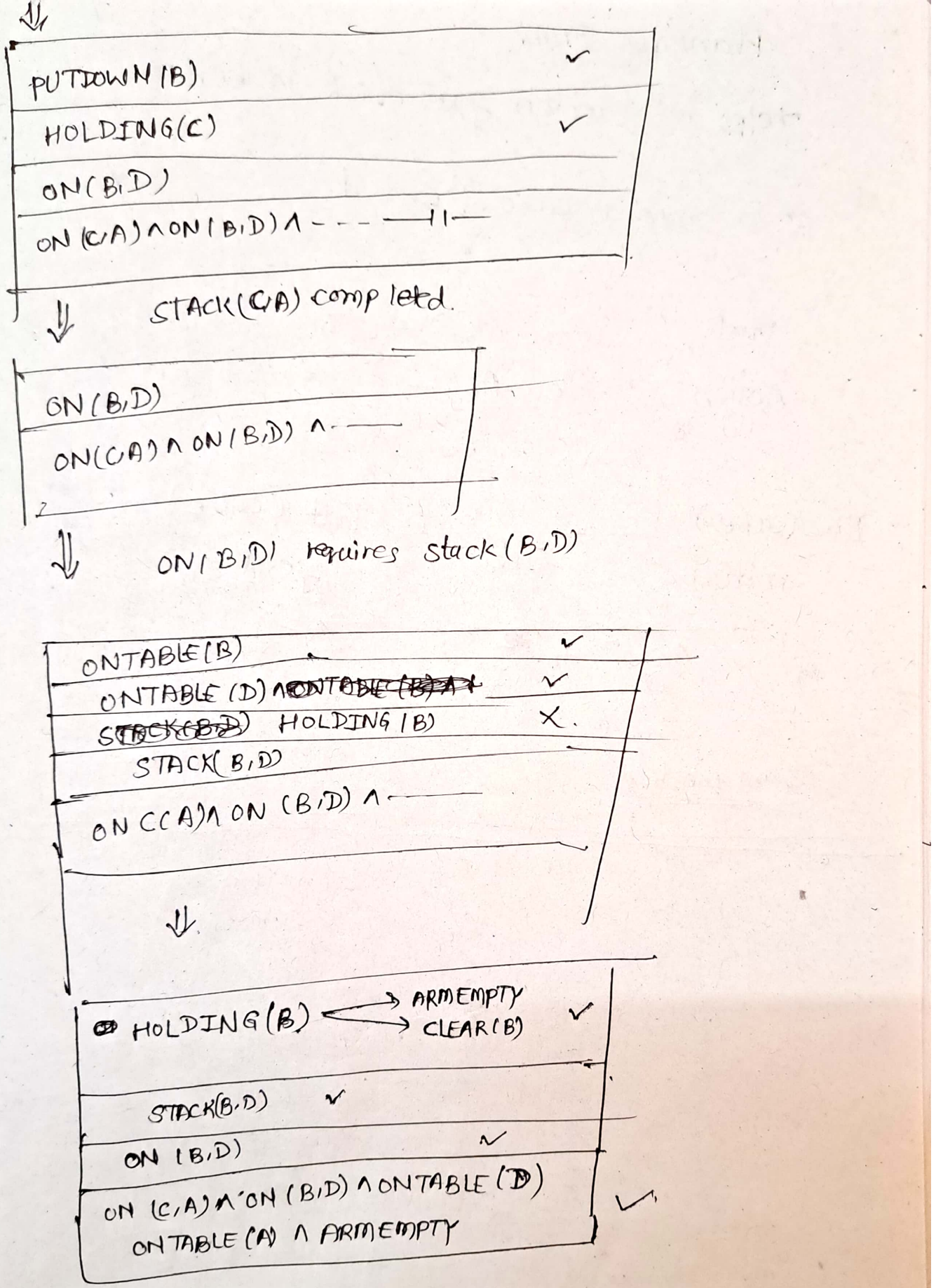
ON(C,A) replaced by stack(C,A)











over

Different goal stack will give different plans

Planning Graph

-helps tell which goal stack is better.

- To have a cake & eat it.

State S_0 S_1

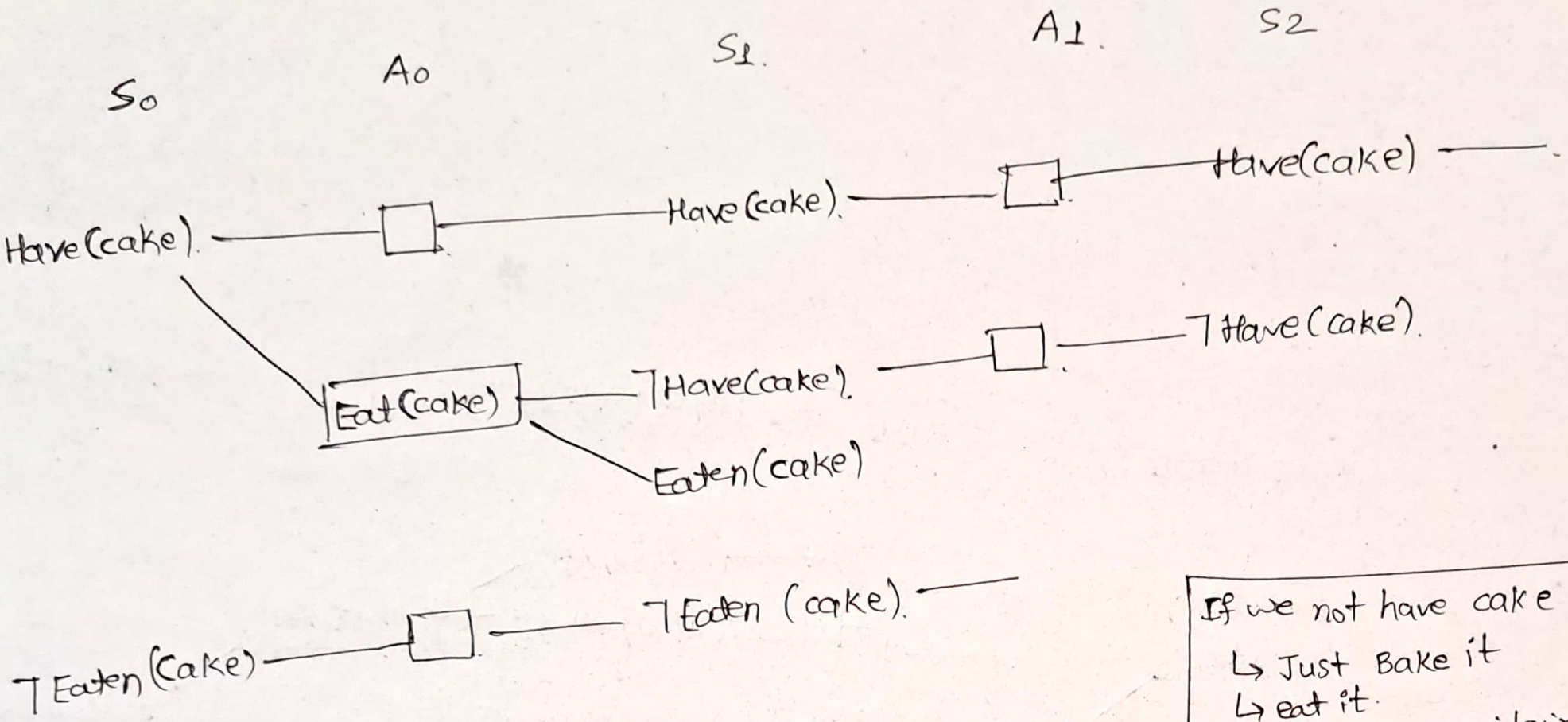
Action A_0 A_1

Have (cake) —Eat→ Eat (cake)

initial goal

$S_0 \xrightarrow[A_0]{} S_1 \xrightarrow[A_1]{} S_2 \rightarrow \cdots \rightarrow S_g$

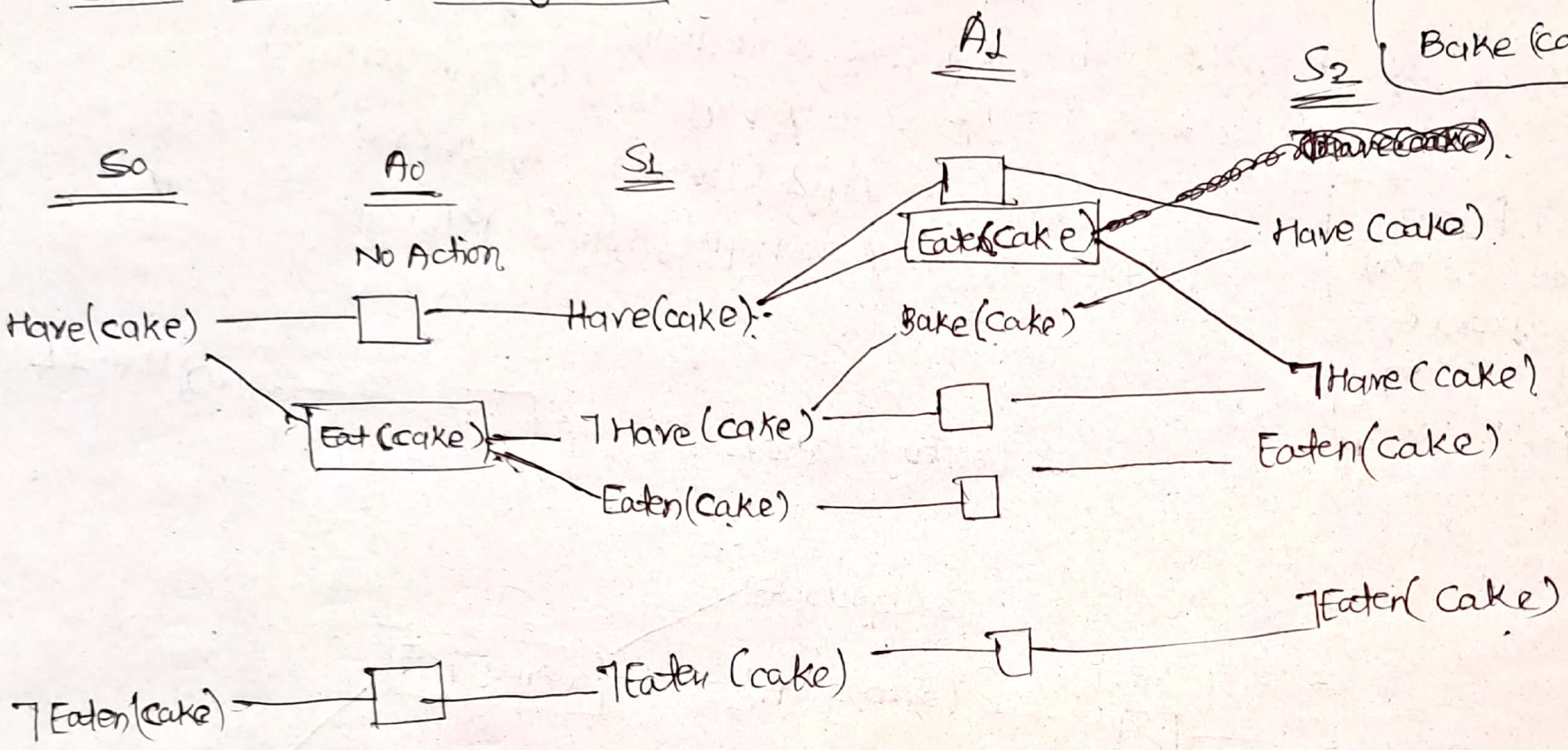
Exam question



If we not have cake
↳ Just Bake it
↳ eat it.
(we are not considering baking here)

- Aim is to find initial state from Goal state.
- This is alternate representation of tree
  polynomial approximation of tree (lesser no. of nodes)
- ~~cannot answer~~ gives guarantee to reach goal state if it exists.
  - estimate no. of ~~goal~~ steps to reach goal.

After considering baking cake:-



★ Pre-conditions of Bake(cake) not mentioned

★ Exam Question - make Tree

Tree Representation

- Action is not mentioned anywhere.

ex:-

Tic-Tac-Toe.

Exam question