Syllabus.

- What is Artificial Intelligence : IA
- Problems, Problem spaces & Search
- Heuristic Search Techniques.
- Knowledge Representation issues.
- Using Predicate logic.
- Representing knowledge using rules.
- Game playing.
- Planning.
- Understanding.
- Natural Language processing.
- Expert Systems.
- AI programming language
  - ↳ prolog, python.

OVERVIEW

Date: 07.06.23

AI: Artificial Intelligence

↳ Simulation of human intelligence in machines.

Problem:

A problem is characterized by a set of goals, a set of objects and a set of operations.

Problem space:

→ feature space.
→ abstract
→ may contain one or more solutions.
→ It could be 1 or 2 or multi-dimensional.

→ Problem space is a process used to solve a problem.

Search:

→ Search for a solution in a problem space.

→ Interclass similarities.
→ Intraclass dissimilarities.

Recognition problem

Production System

Initial state ——————→ Goal state
(Search Operation)

Heuristic Search:

→ Faster than classic methods.
→ Exploration of possible solutions.
→ finds the most acceptable option within a reasonable time limit or within the allocated memory space.

Hill climbing:

→ finds the best available solution by continuing to generate solutions until it

finds the goal state.

Best first Search:

→ a combination of depth first & Breadth first searches.

Problem Reduction:

→ Decomposing the problem into smaller sub-problems.

```
  SEND
+ MORE
--------
 MONEY
   ↓
   1
```

Constraint Satisfaction:

→ where a problem is solved when its values satisfy certain constraint ore rules of the problem.

Knowledge Representation issues:

→ ways are identified to provide machines with the knowledge that human possess so that AI systems can become better.

Using predicate logic:

→ Representing simple facts in logic

Representing knowledge using Rules:

Initial state ⟶ goal state (Forward Reasoning/chaining)

Goal state ⟶ Initial state (Backward Reasoning)

Game playing:

→ Minimax Search Procedure

↳ Recursive/backtracking algorithm

↳ Mostly used for game playing in AI; such as: chess and various two-players game.

↳ Both the players fight it as the opponent player gets the minimum benefit while they get the maximum benefit.

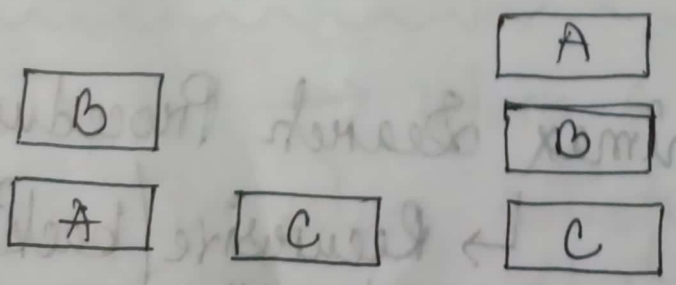
Alpha cut: Represents the maximum value for

Beta cut: " " minimum " "

Planning:

→ Undo করা যাবে না.

→ Blocks world problem.



→ Summarizing the action by rearranging.

Understanding:

Natural language processing:

↳ Spoken word

↳ Written Text

Expert System:

↳ Resolve many issues which generally would requiree a human expert.

↳ Interactive & reliable computer based decision-making system.

↳ Highest level of human intelligence & expertise.

Date: 12.06.23

State space search: Water jug problem

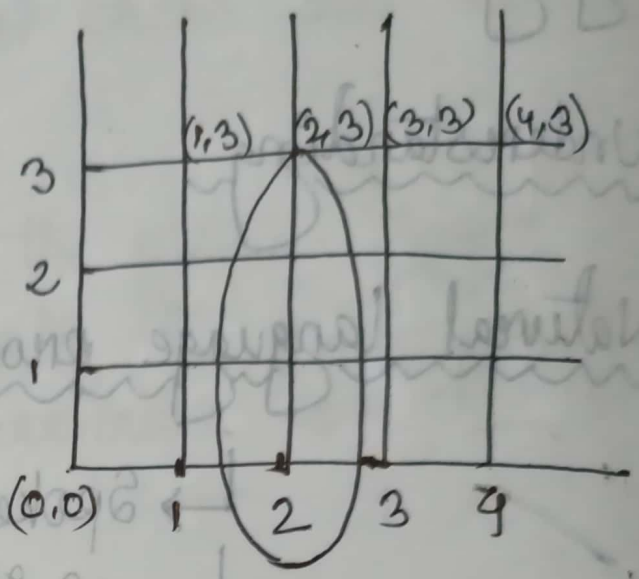
$x \rightarrow$ 1st jug (4L)

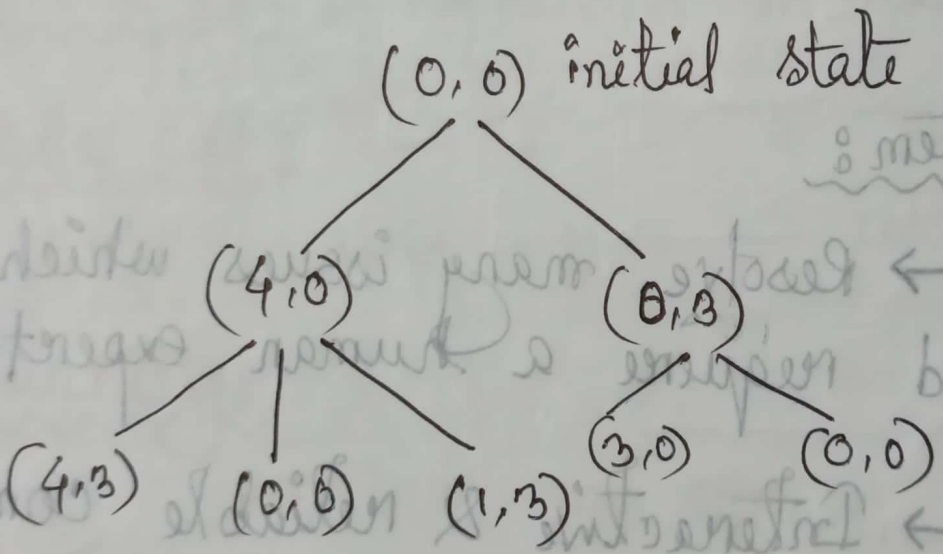
$y \rightarrow$ 2nd jug (3L)

(x, y)

$x = 0, 1, 2, 3, 4$

$y = 0, 1, 2, 3$





Goal state (2, Dx)

① $(x, y) \longrightarrow (4, y)$
if $x < 4$

② $(x, y) \longrightarrow (x, 3)$
if $y < 3$

③ $(x,y) \rightarrow (4, y-(4-x))$
if $(x>0)$

④ $(x,y) \rightarrow (x-(3-y), 3)$
if $(y>0)$

⑤ $(x,y) \rightarrow (0,y)$
if $(x>0)$

⑥ $(x,y) \rightarrow (x,0)$
if $(y>0)$

⑦ $(x,y) \rightarrow (4, y-(4-x))$
if $(x+y \geq 4, y>0)$

⑧ $(x,y) \rightarrow (x-(3-y), 3)$
if $(x+y \geq 3, x>0)$

⑨ $(x,y) \rightarrow (x+y, 0)$
if $x+y \leq 4, y>0$

⑩ $(x,y) \rightarrow (0, x+y)$
if $x+y \leq 3, x>0$

Task domain of AI
* AI Technique

Chapter 2 Date: 14.06.23

Intelligent Agents

→ Sensor : Sense the environment.
→ Actuators: execute.

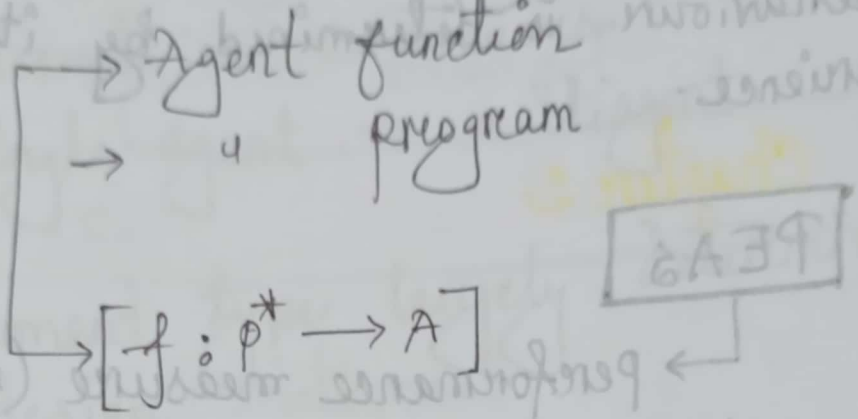
Russhel → book

Agents:

↳ perceive its enviroment through sensors & acting upon that environment through actuators.

→ Human Agent
→ Robotic Agent

Agents and environment:



Vacuum-cleaner world:

→ percepts: location & contents [A, Dirt

→ Actions: left, right, suck, NOOP.

A vacuum cleaner Agent:

→ Rational Agents:

→ Do the right things.

→ maximize its performance measur

performance measure:

Criterion for success of an agent's behaviour.

→ An agent is a) autonomous if its behaviour is determined by its own experience.

PEAS

- → performance measure (safe, fast...
- → Environment (Roads, customers...
- → Actuators (steering wheel, accelerator,
- → Sensors

→ PEAS of a medical diagonstic system.
→ " " part-picking Robot.
→ " " Interactive English tutor

Environment types:

- → Fully observable vs partially observable
- → Deterministic vs stochastic
- → Episodic vs ~~sequir~~ sequential.

↳ Static vs dynamic.
↳ Discrete vs continuous.
↳ Single agent vs multiagent.

→ Environment type largely determines agent design

Agent functions & programs:

↳ percept এর ক্রম agent এর mapping করা।

Table look-up agent:

→ Drawbacks.

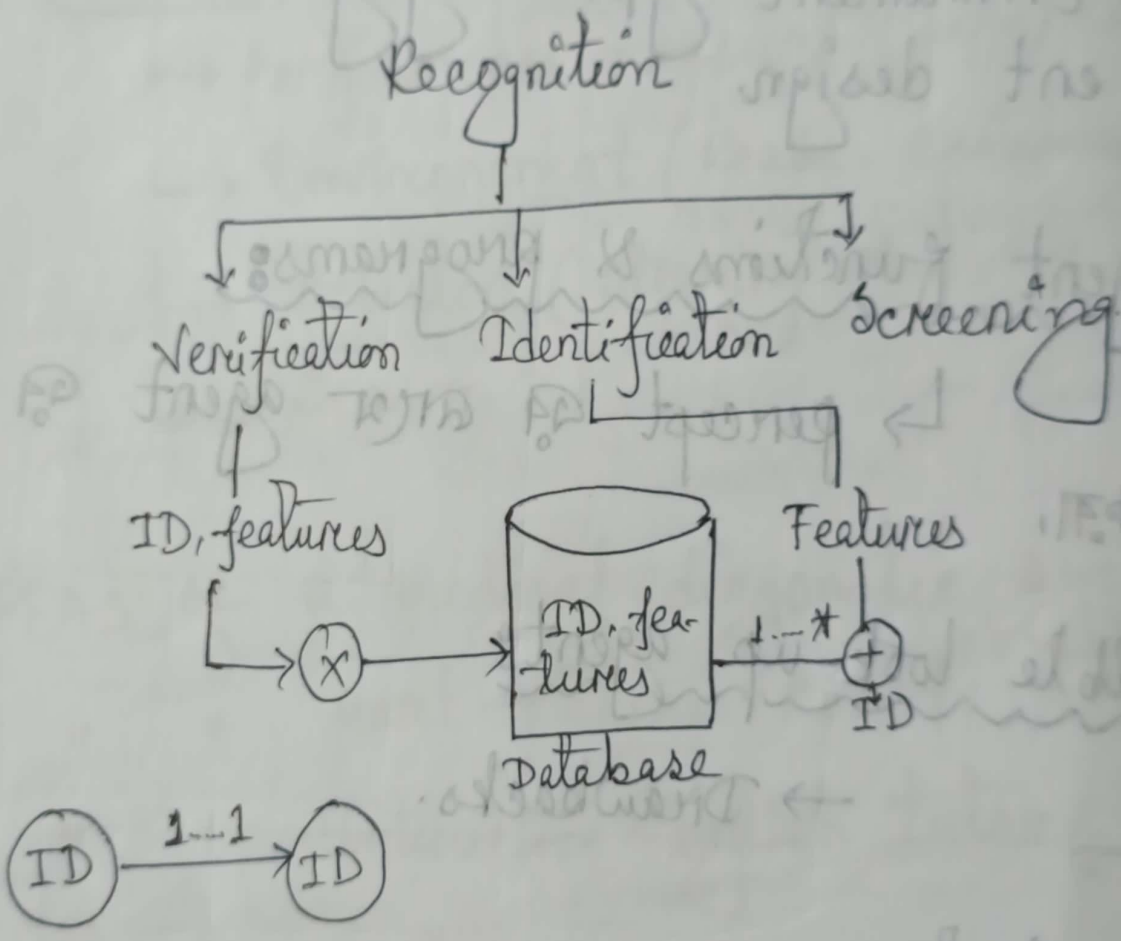
Agent types:

↳ Simple reflex agent
↳ model-based "
↳ Goal "
↳ utility "

Learning agents:

↳ more intelligent

Date: 19.06.23



→ Data Acquisition

→ " Pre-processing

→ Feature Extraction

Width x Height x colour

n = Bits

$\therefore$ colour = 2^n

GIF format → 8 bit

Jpg → 24 "

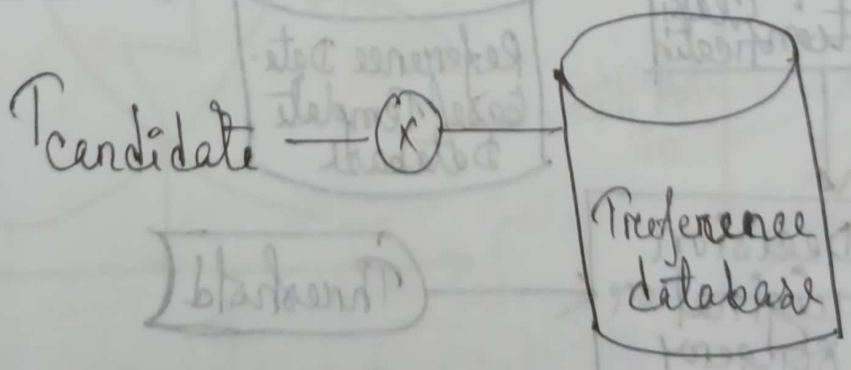
→ Look-up table

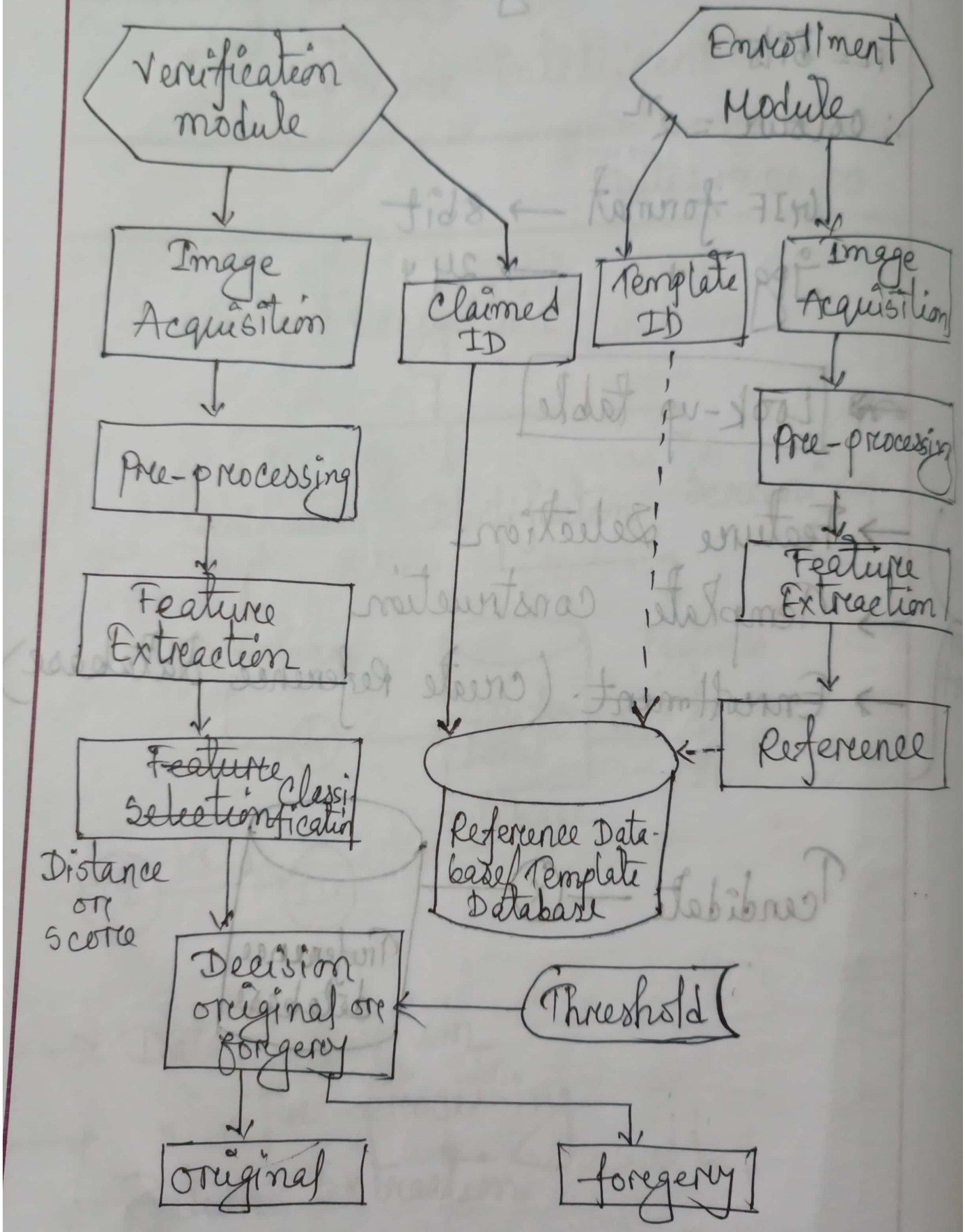
→ Feature Selection.

→ Template construction

→ Enrollment. (create Reference database)

Enroll ment

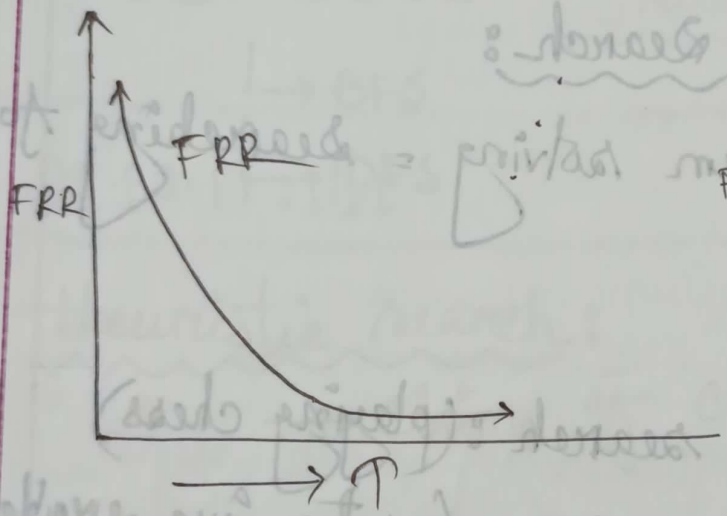


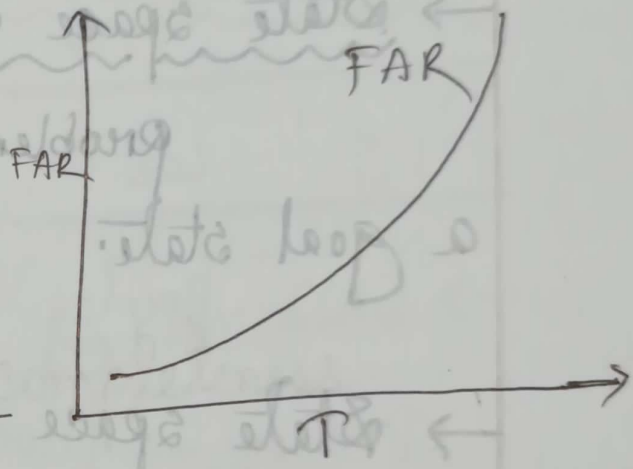

Verification module
Enrollment Module
Image Acquisition
Claimed ID
Template ID
Image Acquisition
Pre-processing
Pre-processing
Feature Extraction
Feature Extraction
Feature Selection
Classification
Reference
Reference Database/ Template Database
Distance or Score
Decision original or forgery
Threshold
original
forgery

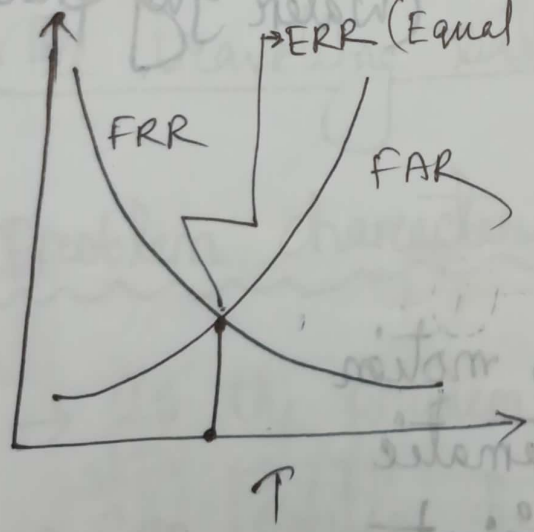
verification errrore

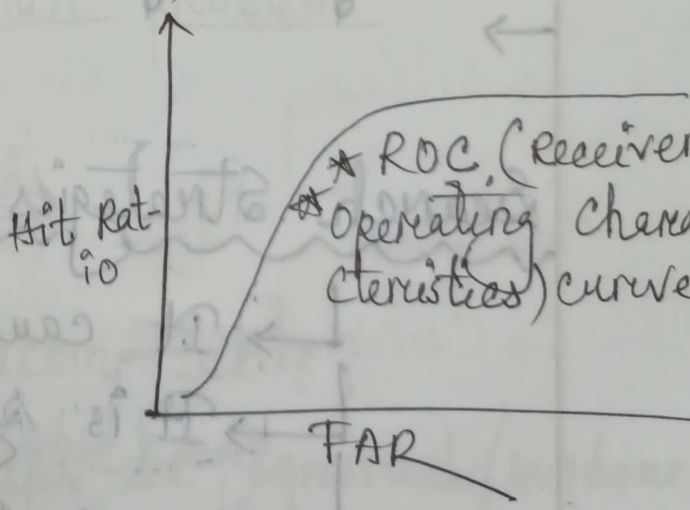
False Rejection Rate (FRR)

False Acceptance Rate (FAR)









→ কোন মডেল better, তা ROC দিয়ে compare করতে হয়।

→ যে System এর area যত বেশি, তত System
তত ভালো,

Problems & Search → বিষদ এই নোটে book

→ **State space Search:**

Problem solving = Searching for a goal state.

→ State space search : (playing chess)

→ " " " (water jug problem)

Search Strategies

- → It causes motion
- → It is systematic
- → " " efficient

→ Uninformed search (Blind search)
→ Informed " (Heuristic "

Blind Search:

→ BFS
→ DFS

Heuristic search:

→ Claims of completeness

The travelling salesman problem

Problem characteristics

→ Is the problem decomposable?
→ Can solution steps be ignored/undone
→ Is the universe predictable?
→ Is a good solution absolute/relative?

→ Is the solution a state / a path?
→ What is the role of knowledge?
→ Does the task requires human-interaction

Problem classification

Lab

Date: 17.07.23

→ SWI Prolog

Prolog → Programming with logic

→ History of prolog.

→ Basic idea of prolog.

→ Consequences.

→ Knowledge Base 1. → Prolog file

```
woman(mia)
woman(jody)
woman(yolanda)
```

playsAirGuitar (mia)
party.

→ Knowledge Base 2

happy (yolanda) ← fact

listens2music (mia) ←

(yolanda) :- happy (yolanda)

playsAirGuitar (mia) :- listens2music (mea)

(yolanda) :- (yolanda)

head — body

Rule

Full stop → clause

No → predicate

→ Expressing conjunction

, → and

; → different

→ Prolog and logic

implication (:-)

Conjunction (,)

Disjunction (;)

→ Knowledge Base 4.

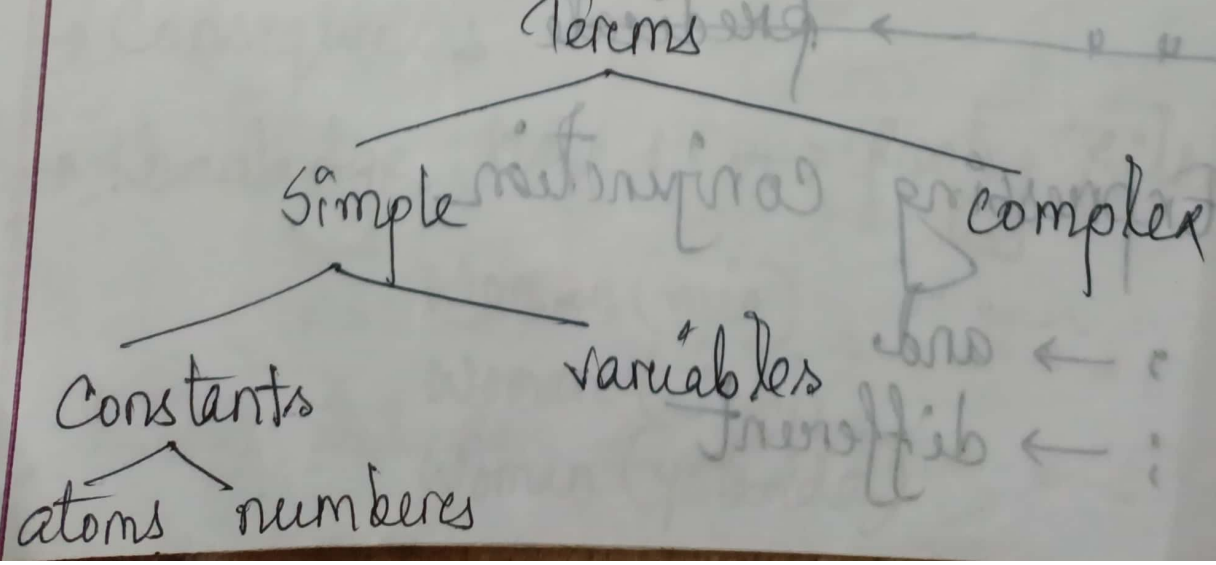
→ Prolog variables

↳ starts with capital letter

→ Constant → small letter

→ Knowledge Base 5.

→ Prolog syntax



Arity:

Number of arguments in a function

lab - 9 MCQ - exam 2141

→ Arity is important.

→ Example:

happy/1

→ Unification in prolog.

Woman(X)

with

Woman(mia)

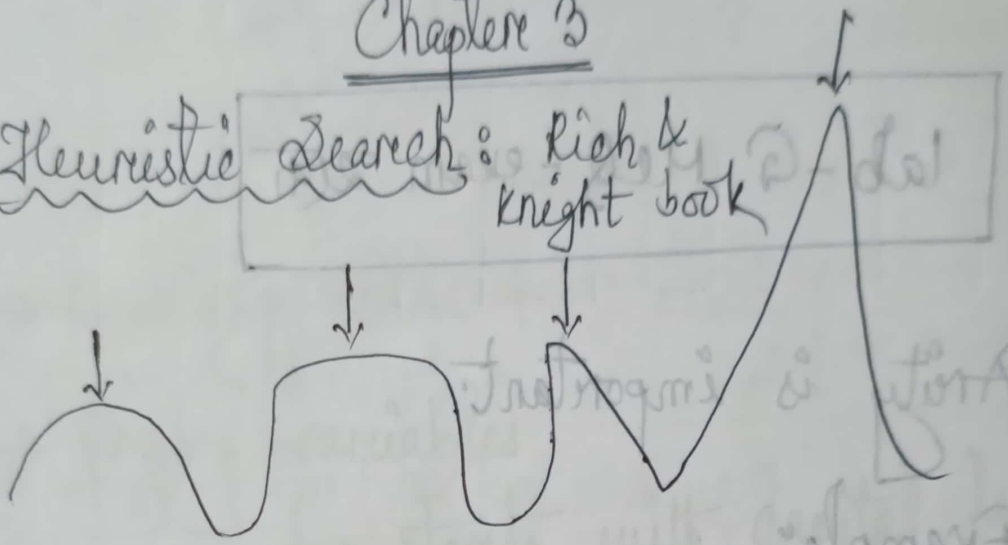
→ Instantiations

→ Occurs check.

→ Proof search

Chapter 3

Heuristic Search: Rich & Knight book



→ Generate-&-test
  - ↳ Exhaustive
  - ↳ Heuristic
  - ↳ plan

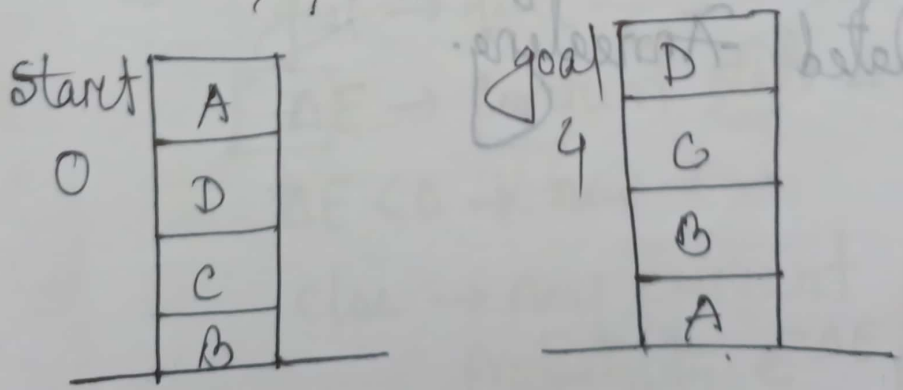
→ Hill climbing
  - goal state = top of a hill
  - generate & test + Direction to move

→ Simple Hill. climbing.

→ Steepest - ascent Hill climbing.
(Gradient Search)

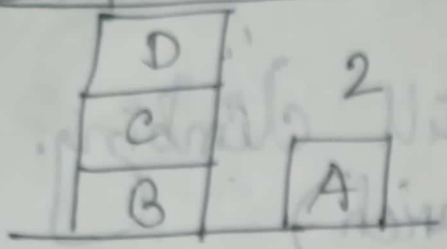
→ Hill climbing (Disadvantages)
→ local maximum
→ plateau
→ Ridge

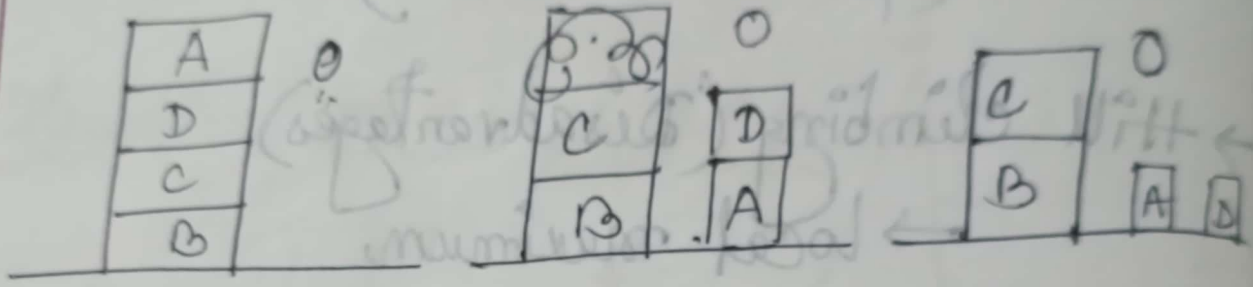
Ways out:
→ Backtrack
→ Big Jump
→ Several directions



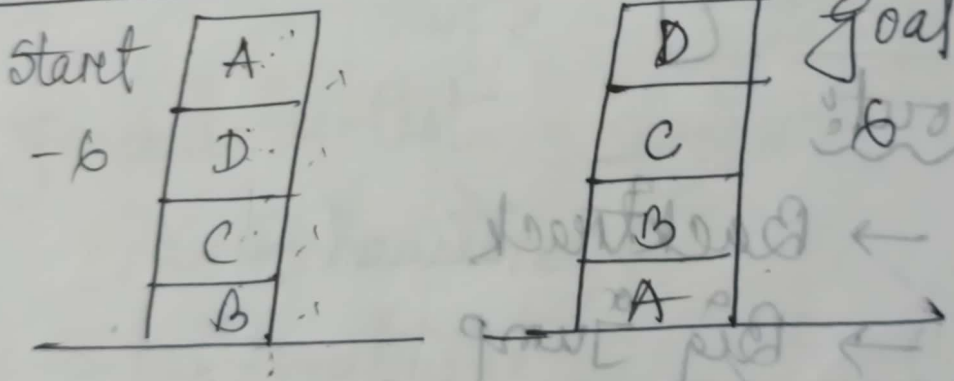
Blocks world

Local Heuristic :

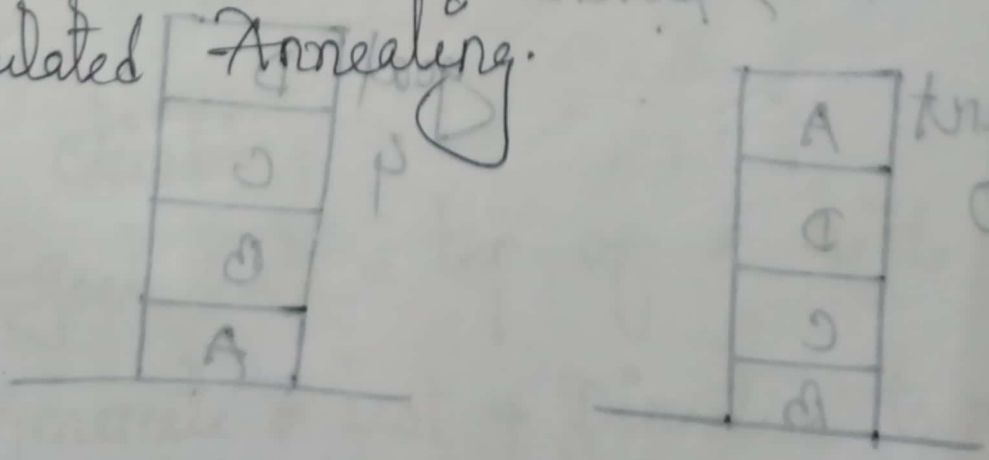




Global Heuristic :



→ Simulated Annealing.



Date: 26.07.23

Simulated Annealing:

- → downhill moves may be made
- → enough exploration of the whole shay space
- → lowering the chances.

→ Physical Annealing:

- → minimal energy.
- → gradually cooled
- → Annealing ~~sh~~ schedule
- → $e^{-\Delta E/kT}$

Algorithm

goal → quit

ΔE → Current state − new state

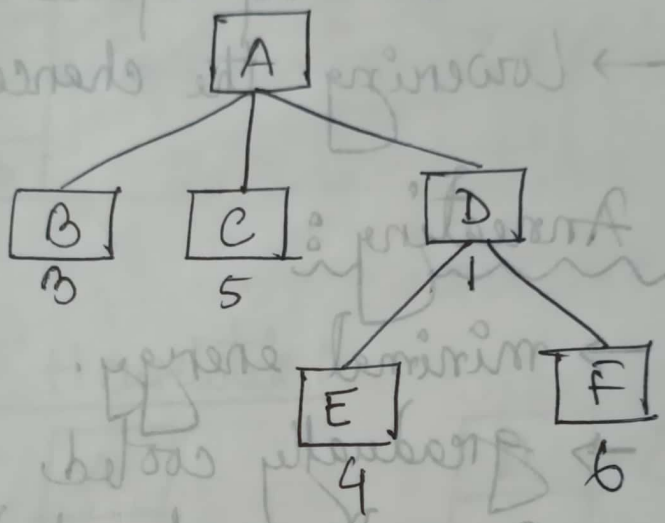
$\Delta E < 0$ → new

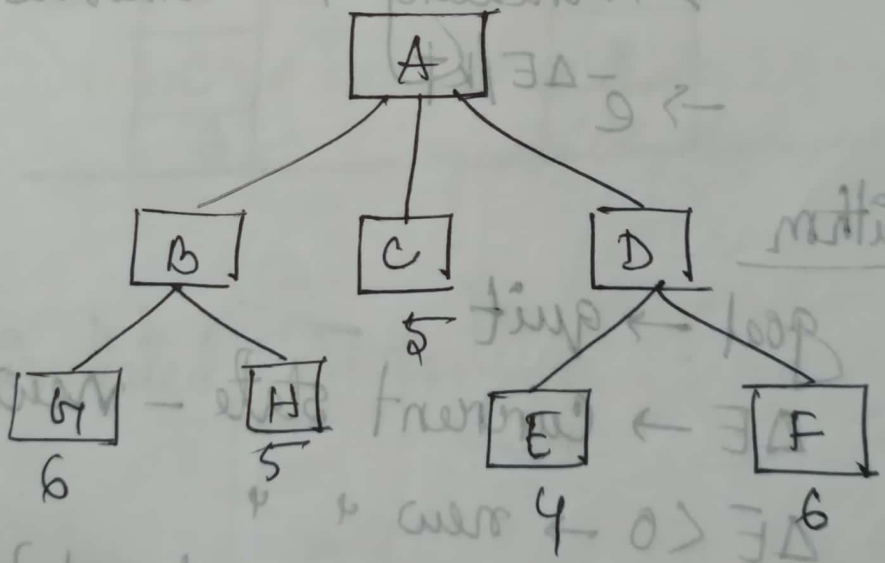
else → new current state with probability $e^{-\Delta E/kT}$.

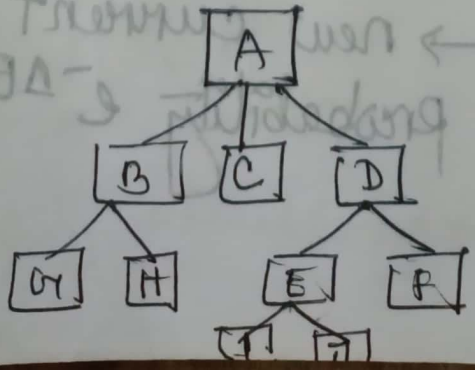
Best-First Search:

↳ Breadth - First search

↳ Depth -







OPEN:

→ priority queue

→ nodes not examined

CLOSED:

→ nodes have already been examined

→ Check whether it generated before

Algorithm

Greedy Search → neither optimal nor complete

Uniform Cost Search

↳ optimal & complete, very inefficient

Algorithm A*: (Rich & knight)

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

↓ depth factor

↓ heuristic factor

* Algorithm, exam-G আসবে।

→ Real cost
→ estimated cost

Problem Reduction: (Rich & Knight).

Algorithm AO* → modified A*

Contraint satisfaction:

```
  SEND
+ MROE
-------
 MONEY
```

⇒ cryptoarithmatic puzzle.

```
  SEND
+ MORE
------
 MONEY
 ↑ ↑ ↑
 1 O E+1
```

```
   9
   1 0
+  1 0
------
   1 0
```

Russel

chapter : 3

30.09.23

Function : SIMPLE PROBLEM SOLVING AGENT

Static :

seq
state
goal
problem

state ← UPDATE-STATE (state, percept)

goal ← FORMULATE-GOAL (state)

problem ← " - PROBLEM (state, goal)

seq ← SEARCH (problem)

action ← FIRST (seq)

seq ← REST (seq)

return action

Example: Romania

Problem types:

→ Deterministic, fully observable

→ Non-observable

→ Non-Deterministic, partially

→ Unknown state space

Example: Vacuum world

→ Vacuum world state space graph

Example: The 8-Puzzle

Example: Robotic Assembly

→ Tree Search algorithm

→ Implementation : general Tree Search

→ " : States vs. nodes

State → physical config

node → data structure
- → State
- → parent node
- → action
- → path cost.

→ Search Strategies.
- → Completeness
- → time complexity
- → Space
- → optimality.

→ Depth-First এ infinite loop এ যাওয়ার possibility থাকে।

b d m

Uninformed Search strategy :

↳ BFS

↳ DFS

↳ Uniform Search → BFS এর মত কাজ করে।

→ Properties of BFS

→ ৭ ৭. DFS

Depth-limited search :

↳ level limit দেয়া থাকে।

→ Iterative deepening search

→ Repeated States

Chapter-4 (Russel) Date: 30.09.23

BFS → Best-First Search

→ Greedy - Best First Search

$f(n) = h(n)$

→ Properties of greedy-BFS.

→ A* Search

Ques
→ Graph যেখানে একবার Cost কোনটার থেকে কোনটা কত যেখানে হবে।
↳ Algorithm যেখানে হবে।

Admissible heuristics:

$h(n) \leq h^*(n)$

- $h(n)$: estimated cost
- $h^*(n)$: Real cost

∴

→ optimality of A^* (proof)
skip

→ Hill climbing Search

Chapter : 04 (slide) Rich Date: 04.09.23

Knowledge Representation

→ What is KR?
  ↳ onthology

→ Facts

→ Representation of facts.

Representation & mapping:

Spot is a dog.

Every dog has tail.

∴

Spot has a tail.

dog (spot)

$\forall x$: dog(x) → hastail (x)

hastail (spot)

→ The multilated checkereboared problem

→ Good representation

→ Approaches to KR

↳ Simple relational knowledge

↳ Inheritable knowledge
↳ Inferential
↳ procedural

→ choosing the granularity.

Chapter-5

→ Using propositional Logic
predicate

→ ① Theorem proving is decidable
② Can't represent objects & quantification.

① Theorem proving is semi-decidable
② Can represent objects & quantifica

or

→ Conversion to clause form

→ Reasoning.

→ Resolution

→ ↳ in propositional logic

↳ Unification

1. $P \rightarrow Q = \neg P \vee Q$

2. $(\neg P \vee Q) = \neg P \vee \neg Q$

3. $\neg(P \wedge Q) = \neg P \wedge \neg Q$

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

$\neg \neg P = P$

4. $(\forall x : P(x)) \vee (\exists x : Q(x)) = (\forall x : P(x)) \vee (\exists y : Q(y))$

→ Skolem

→ Algorithm : Resolution

Chapter 6

→ Declarative vs procedureal knowledge
- Declarative → not embedded
- procedureal → embedded

→ Logic Programming

→ Forward vs Backward Reasoning.
- Forward → From start states
- Backward → From goal states

→ Why better to reason forward to backward ?

↳ Four factors

→ Production System
  ↳ Working memory
  ↳ Rule

→ RETE Networking

→ Matching
  ↳ ELIZA

→ Conflict Resolution
  ↳ Preference based on rules
    ↳ objects
    ↳ status

chapter 8 (Rich)
Game playing

→ minimax search procedure

→ Alpha cut, Beta cut

→ generate procedure
  → Test
  → plausible move vs legal move

→ Minimax search
  → Depth first search
  → limited

→ Two ply search
  → maximizing ply (player)
  → Minimizing " (opponent).

Player (position, Depth):
Opponent (position, Depth):
Any-player (" , "):

MINIMAX (position, Depth, player):
MOVE-GEN (position, player)
STATIC (" , ")
DEEP-ENOUGH (" , depth)

→ Adding alpha, beta cut offs.
  ↳ maximize (> α threshold)
  ↳ Minimize (< β ")

Rich book: Alpha-Beta cut off example

→ Additional Refinements

- ↳ futility cutoffs
- ↳ quiescence
- ↳ Secondary search.

→ Iterative Deepening

Chapter-13

Planning

P → precondition

D → Delete

A → Action

STACK (x,y)

UNSTACK (x,y)

PICK UP (x)

PUTDOWN (x)

PICK UP
PUT DOWN } Table

STACK
UNSTACK } Block.

→ Goal Stack planning

→ Sussman Anomaly problem

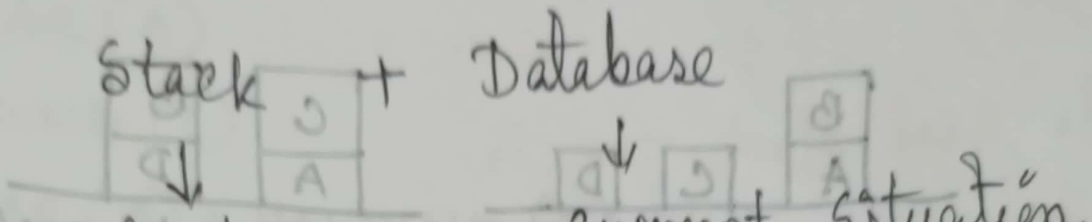
assignment सेट एवं अभ्यास (Rich)

planning:

problem solving in advance

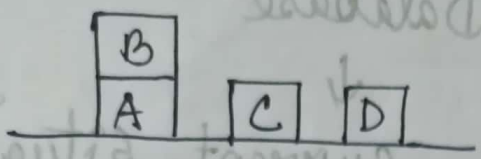
→ The blocks world

Goal stack planning:

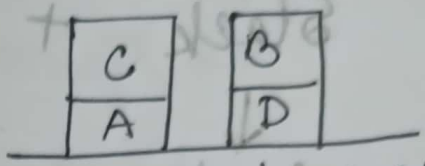


Assignment 11.09.23

Goal stack planning:



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



goal: ON(C,A) ∧
~~ONTABLE(A)~~ ∧
ON(B,D) ∧
ONTABLE(A) ∧
ONTABLE(D)

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

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

STACK(C,A)
ON(B,D)
ON(C,A) ∧ ON(B,D) ∧ OTAD

CLEAR (A) ∧ HOLDING (C)

CLEAR (A)

HOLDING (C)

CLEAR (A) ∧ HOLDING (C)

STACK (C, A)

ON (B, D)

ON (C, A) ∧ ON (B, D) ∧ ONTAD

ON (B, A)

CLEAR (B)

ARMEMPTY

ON (B, A) ∧ CLEAR (B) ∧ ARMEMPTY

UNSTACK (B, A)

HOLDING (C)

CLEAR (A) ∧ HOLDING (C)

STACK (C, A)

ON (B, D)

ON (C, A) ∧ ON (B, D) ∧ ONTAD

ONTABLE(A) ∧ ONTABLE(C) ∧ ONTABLE(D) ∧
HOLDING(B) ∧ CLEAR(A)

The goal stack now is,

HOLDING(C)
CLEAR(A) ∧ HOLDING(C)
STACK(C,A)
ON(B,D)
ON(C,A) ∧ ON(B,D) ∧ ONTAD

Two branches of the search tree corresponding to the following goal stacks:

(1)
ONTABLE(C)
CLEAR(C)
ARMEMPTY
ONTABLE(C) ∧ CLEA(C) ∧
ARMEMPTY
PICKUP(C)
CLEAR(A) ∧ HOLDING(C)

STACK (C,A)

ON (B,D)

ON (C,A) ∧ ON(B,D) ∧ OTAD

(2) ON (C,x)

CLEAR (C)

ARMEMPTY

ON(C,x) ∧ CLEAR(C) ∧
ARMEMPTY

UNSTACK (C,x)

CLEAR (A) ∧ HOLDING(C)

STACK (C,A)

ON(B,D)

ON(C,A) ∧ ON(B,D) ∧ OTAD

To satisfy ON(C,x), we would have to STACK C onto some block x, The goal stack then would be,

CLEAR(x)

HOLDING (C)

CLEAR(x) ∧ HOLDING(C)

STACK (C, x)

CLEAR (C)

ARM EMPTY

ON(C, x) ∧ CLEAR(C) ∧ ARMEMPTY

UNSTACK (C, x)

CLEAR (A) ∧ HOLDING(C)

STACK (C, A)

ON(B, D)

ON(C, A) ∧ ON(B, D) ∧ ONTAD

We choose to Apply STACK (B, D) by binding D to x in the Stack operator. This makes the goal stack:

CLEAR (D)

HOLDING (B)

CLEAR (D) ∧ HOLDING (B)

STACK (B, D)

ONTABLE(C) ∧ CLEAR(C) ∧ ARMEMPTY

PICKUP(C)

CLEAR(A) ∧ HOLDING(C)

STACK(C,A)

ON(B,D)

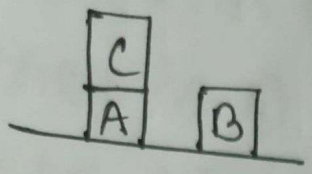
ON(C,A) ∧ ON(B,D) ∧ ONTABLE

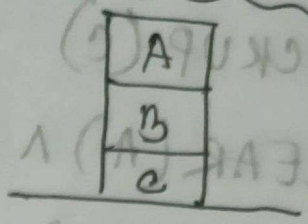
ONTABLE(A) ∧ ONTABLE(C) ∧ ONTABLE(D) ∧
ON(B,D) ∧ ARMEMPTY

The problem solver can now halt & return as ~~the~~ its answer the plan.

1. UNSTACK(B,A)
2. STACK(B,D)
3. PICKUP(C)
4. STACK(C,A)

Goal Stack planning: (Sussman anamoly)





Start: ON(C,A) ∧
ON TABLE(A) ∧
ARM EMPTY

Goal: ON(A,B) ∧
ON(B,C)

ON(C,A)

CLEAR(C)

ARMEMPTY

ON(C,A) ∧ CLEAR(C) ∧ ARMEMPTY

UNSTACK(C,A)

ARMEMPTY

CLEAR(A) ∧ ARMEMPTY

PICKUP(A)

CLEAR(B) ∧ HOLDING(A)

STACK(A,B)

ON(B,C)

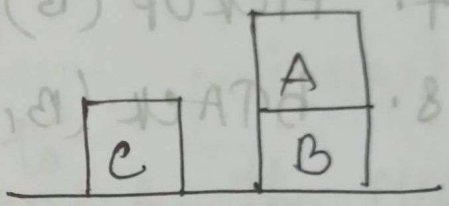
ON(A,B) ∧ ON(B,C)

We can continue popping until the goal stack is

ON(B,C)
ON(A,B) ∧ ON(B,C)

Then the current state is

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



This problem is often called the Sussman Anamoly.

The sequence of operators applied so far is

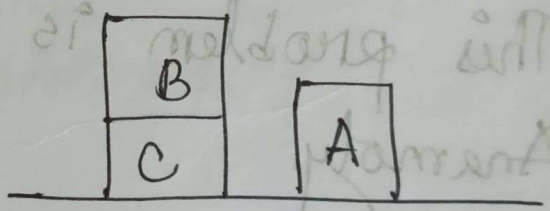
1.) UNSTACK(C,A)
2) PUTDOWN(C)
3.) PICKUP(A)
4.) STACK(A,B)

By the time, we achieved the goal ON(B,C) and popped it off the Stack, we will have executed the following:

5. UNSTACK (A, B)
6. PUTDOWN (A)
7. PICKUP (B)
8. STACK (B, C)

The problem state will be

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



This time,

9. PICKUP (A)
10. STACK (A, B)

The complete plan:

1.) UNSTACK (C, A)
2.) PUTDOWN (C)
3.) PICKUP (A)
4.) STACK (A, B)
5.) UNSTACK (A, B)
6.) PUTDOWN (A)
7.) PICKUP (B)
8.) STACK (B, C)
9.) PICKUP (A)
10.) STACK (A, B)

Eliminating Step 3 ~~and~~ to 6, the resulting plan is.

1.) UNBSTACK (C, A)
2.) PUTDOWN (C)
3.) PICKUP (B)
4.) STACK (B, C)
5.) PICKUP (A)
6.) STACK (A, B)

Chapter-18 11.09.23

Learning from Observations (Russel)

→ Induction learning

→ Hypothesis

→ Learning

→ learning agents

→ learning elements

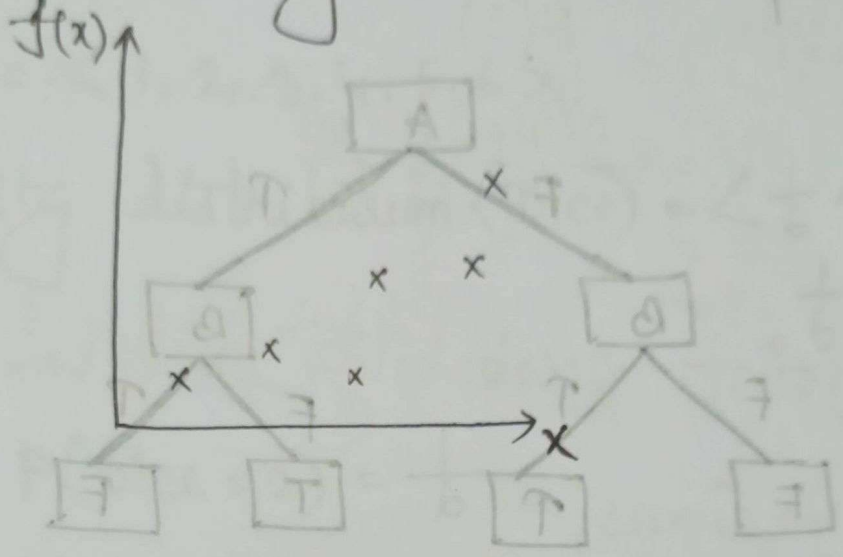
Type of feedback:

- → Supervised learning
- → Unsupervised "
- → Reinforcement "

→ Inductive learning

$(x, f(x))$

→ Inductive learning method



→ Ocham's razore

↳ prefere the simplest hypothesis.

→ Learning decision trees

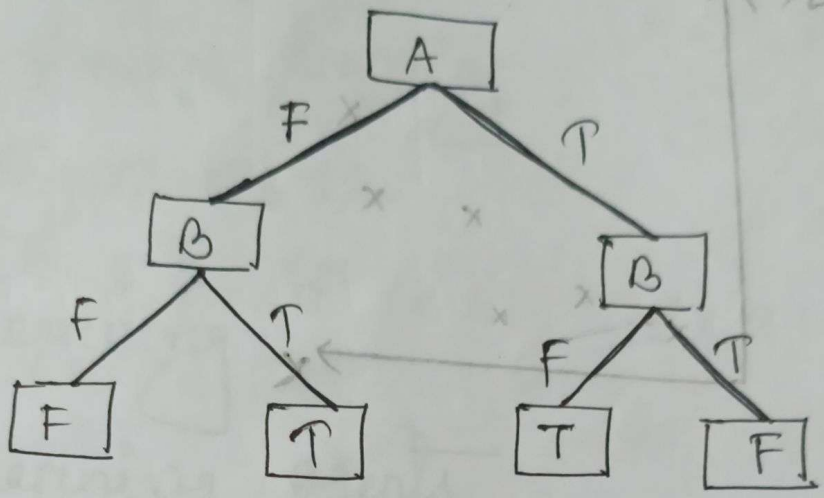
↳ প্রতিটা leaf একটা decision

↳ entropy value কম হলে সেটাকে proceed করাও হয়।

→ Attribute based representation

→ Decision trees.

→ Expressiveness



→ Hypothesis spaces

2^n rows $= 2^{2^n}$

→ Decision tree learning

→ performance measurement.

chapter-8 (Rich)

→ probability = a degree of belief.

→ prior = known

random variable = Dice

domain = $\langle 1, 2, 3, 4, 5, 6 \rangle$

probability distribution (Dice) = $\langle \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6} \rangle$

$$P(Dice = 2) = \frac{1}{6}$$

$$P(A|B) = P(A \wedge B) / P(B)$$

$$P(A \wedge B) = P(A|B) \cdot P(B)$$

→ Conditional probability

S = Stiff neck

M = Meningitis

$P(S|M) = 0.5$

$P(M) = 1/50000$

$P(S) = 1/20$

$$P(M|S) = P(S|M) \cdot P(M) / P(S) = 1/5000$$

$$P(M|S) = \frac{P(S \wedge M)}{P(S)}$$

$$= \frac{P(S|M) \cdot P(M)}{P(S)}$$

→ Joint probability distribution

→ Axioms

$P(\text{true}) = 1$

$P(\text{false}) = 0$

$P(A \vee B) = P(A) + P(B) - P(A \wedge B)$

→ Derived properties

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

$P(U) = P(A_1) + \ldots + P(A_n)$

$U = A_1 \vee A_2 \vee A_3 \ldots$ collectively exhaustive

~~$A \wedge B$~~

→ Bayes' Theorem

→ Id Independence → $P(A \wedge B) = P(A) \cdot P(B)$

$P(A) = P(A|B)$

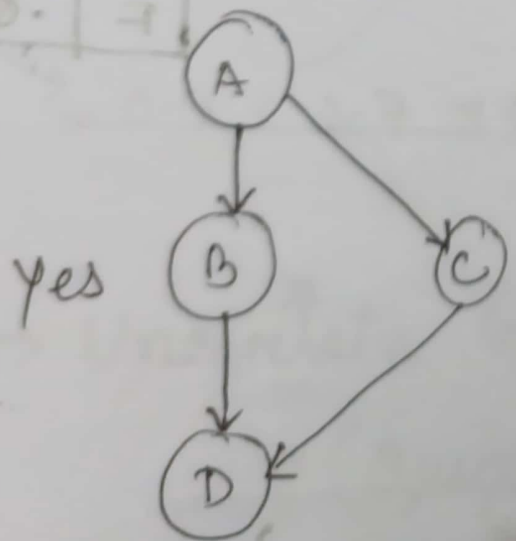
→ Conditional → $P(A \wedge B|E) = P(A|E) \cdot P(B|E)$

→ probability — John and Mary (Example ~~Math~~)

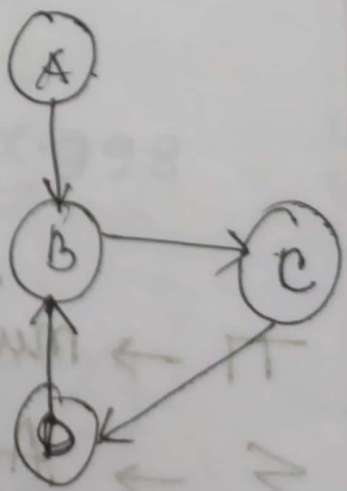
↳ exam - G

→ Bayesian Network

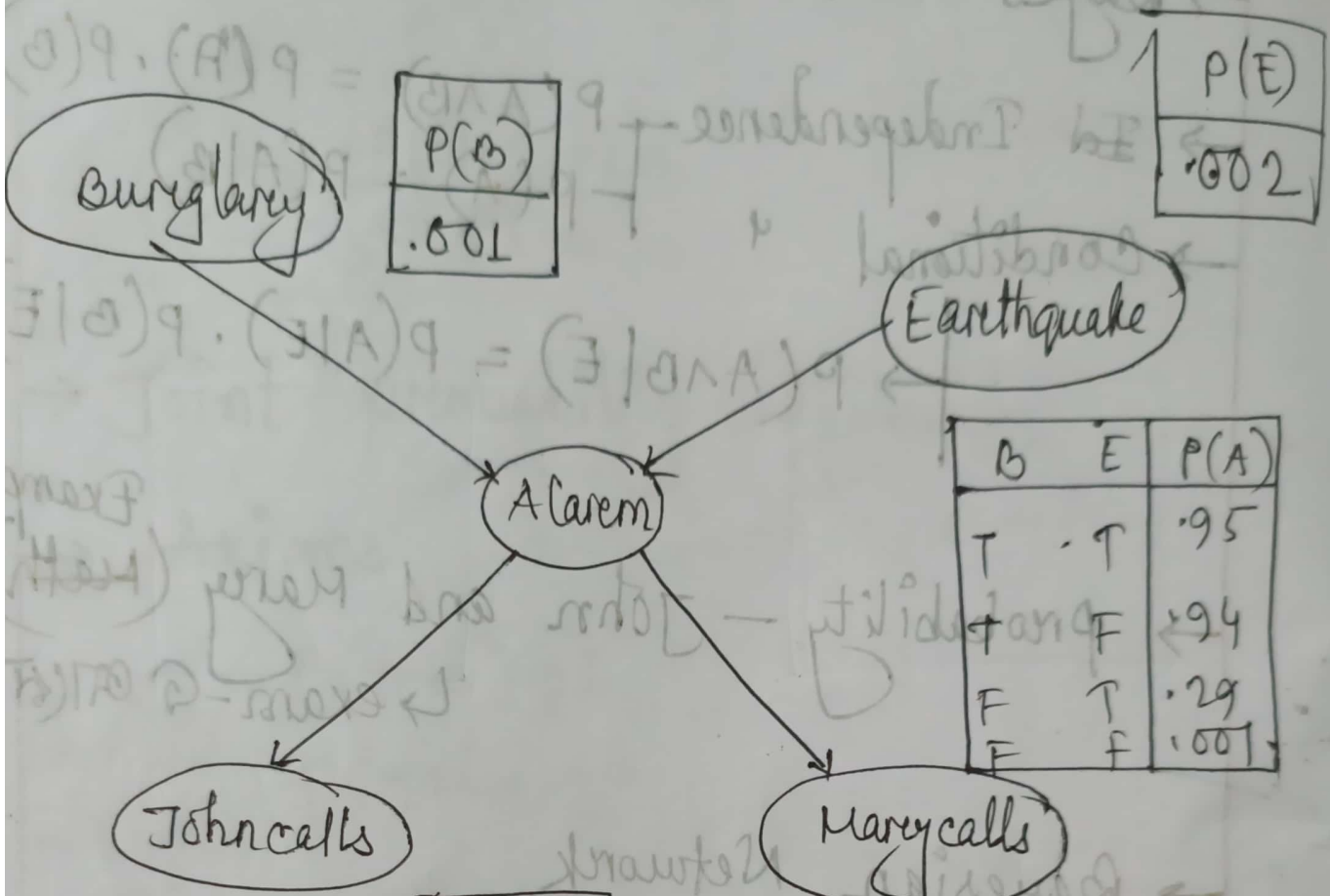
→ syntax



yes



NO

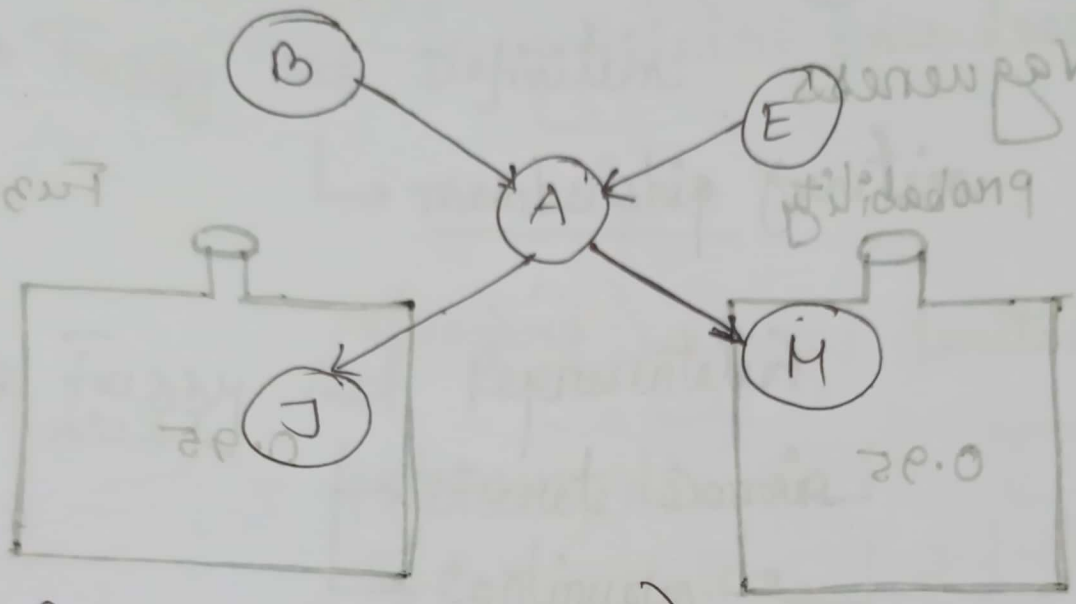


Π → multiplication

Σ → Addition / sum

** What is Syntax & semantics. Describe

→ Semantics



$P(J \wedge M \wedge A \wedge \neg B \wedge \neg E)$

$= P(J|A) \cdot P(M|A) \cdot P(A|\neg B \wedge \neg E) \cdot P(\neg B) \cdot P(\neg E)$

$= 0.00062$

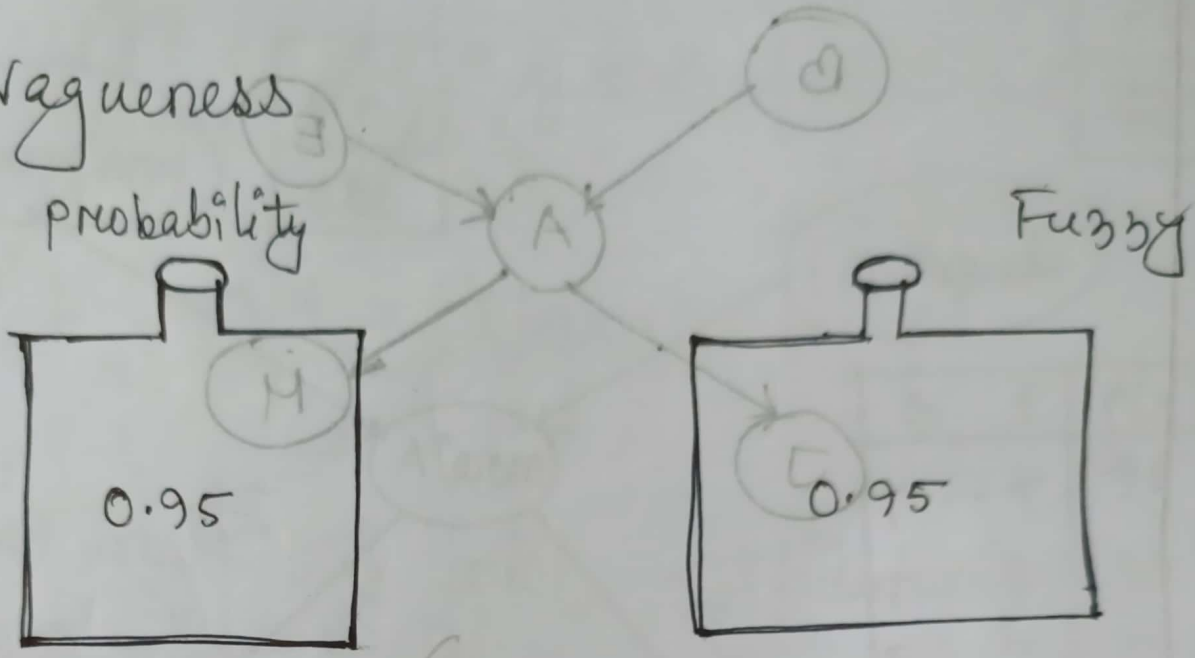
$= 0.9 \times 0.7 \times 0.001 \times .999 \times .998$

→ Uncertain Question Answering

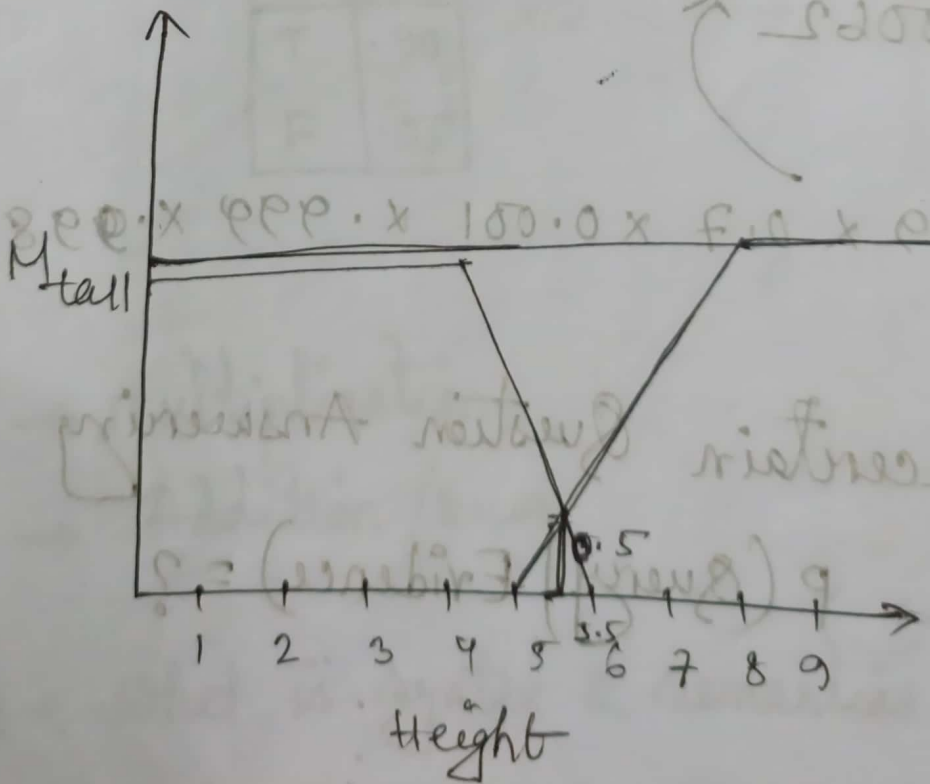
- P(Query | Evidence) = ?

→ General Conditional Independence

→ Vagueness



→ Difference betn. probability & fuzzy.



→ Imprecision Vs. Uncertainty

→ Fuzzy set Definition

  ↳ membership function

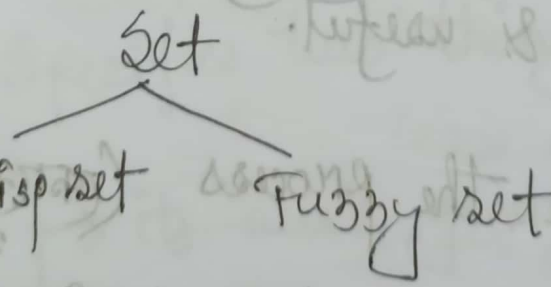
→ Fuzzy set Representation

  ↳ Discrete Domain

  ↳ Continuous ৭

  ↳ α-cuts

→ Fuzzy controller



→ Fuzzy controller ৭? component কিকিঃ
(Ques)

20.09.23

→ Forward chaining
→ Backward chaining (Russel book)

Chapter-15

→ Natural Language processing (Rich book)
  ↳ Speech processing
  ↳ Text ৭

→ Entire language processing

→ Features of languages that make it both difficult & useful.

***→ Steps in the process (কোনো Sentence বোঝা বানাতে)
  ↳ Morphological Analysis
  ↳ Syntactic ৭
  ↳ Semantic ৭

↳ Discourse integration
↳ pragmatic Analysis.

'I eat rice.

→ Sytactic processing
↳ Two main componets of parsing.
↳ grammer (declarative)
↳ parsers (procedural)

→ Grammers & parsers
↳ Specifies two things about a language
↳ its weak generative capacity.
↳ " Strong " " .

→ Parse tree.

→ Kind of parsers
- → Top-down → forwarded
- → Bottom-up → Backward

→ Bottom-up parsing with Top-down filtering.

→ Finding one interpretation / finding many.

→ Four ways of handling sentences.
- → All paths
- → Best path with Backtracking
- → Best
- → wait & see, patch-up

→ Augmented Transition Networks.

→ Semantic Analysis

↳ Lexical processing

↳ Semantic markers

↳ physical-object

↳ Animate - 4

↳ Abstract - 4

→ Lexical disambiguation.

→ Sentence level processing

↳ Semantic grammers ***

→ principal advantages of semantic grammers.

→ Drawbacks to use the semantic greammers.

→ ~~Disab~~ Discourse & pragmatic processing.

→ Speech acts

→ Conversational postulates.

Out of Syllabus:

→ Trend of AI

→ Open AI, generative AI

Chapter-6 (lab) পড়ে নিতে হবে।

lab quiz, slide থেকে আসবে।

Notebook-25

Attendance-25

quiz-5

final lab & viva-5

lab এ practice - বাকি মার্কস

→ class এ যা বলো-10

Book- Prolog programming in Depth.

↳ part II (Application)

↳ A simple expert system shell

↳ An expert system with uncertainty.

↳ natural language processing.

Turbo prolog book

↳ program 22 & 52 * (class-এ করতে বলেছিলো।

↳ Hardware Simulation

↳ program 54 *

↳ Towers of Hanoi

↳ program 55 *

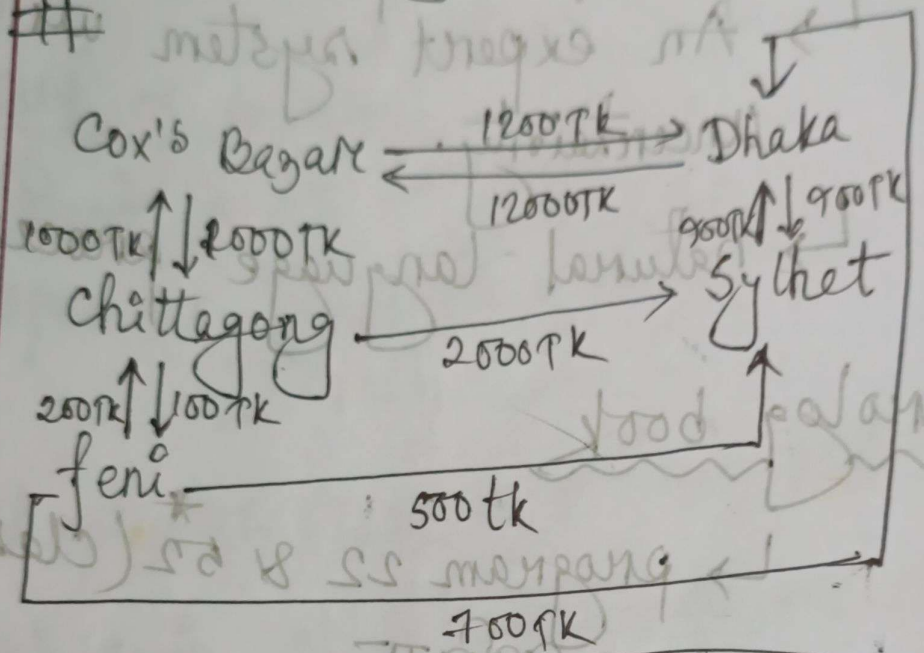
↳ Division of words into symbols.

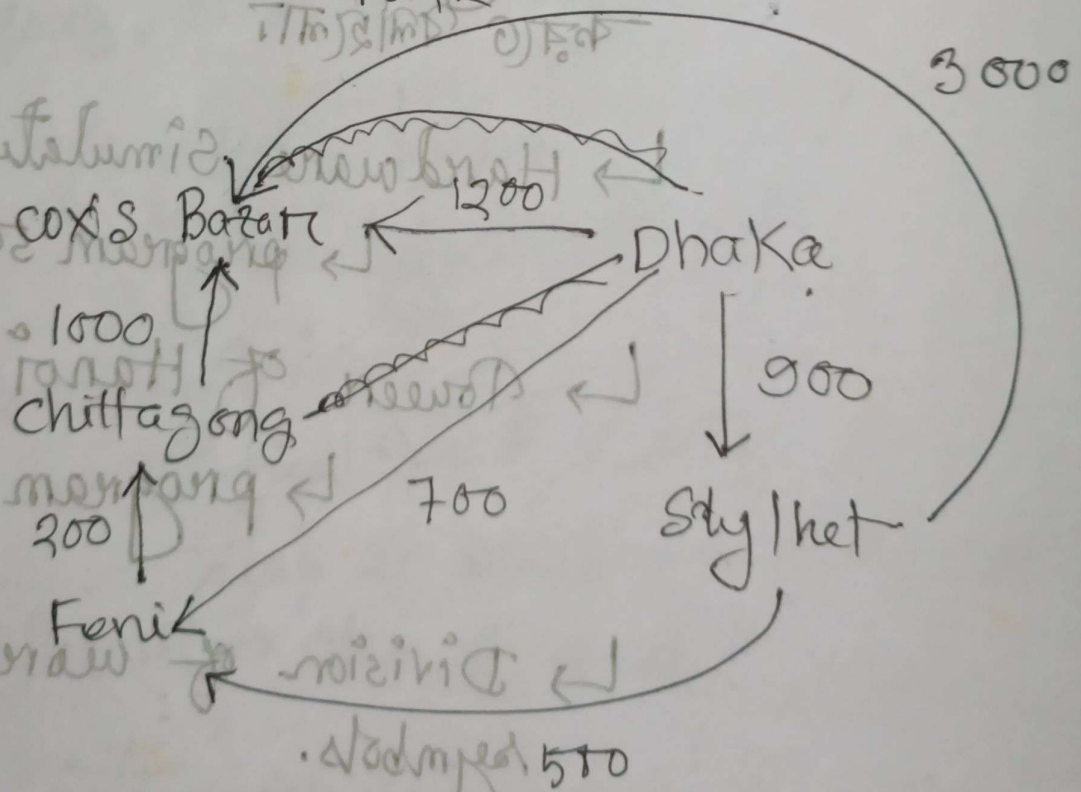
↳ program 56 *

~~26~~. 25.09.

→ Lab evaluation

↳ program 52, 54, 55, 56.





Artificial Neural Networks: A Tutorial.

→ Tasks that neural network can perform.

- ↳ pattern classification
- ↳ clustering.
- ↳ function approximation
- ↳ prediction / forecasting
- ↳ optimization.
- ↳ content-addressable memory.

→ Why Artificial neural network?

- ↳ massive parallelism.
- ↳ distributed representation & computation.
- ↳ learning ability
- ↳ generalization ability.

↳ adaptivity
↳ inherent contextual info processi
↳ fault tolerance
↳ low energy consumption.

→ Biological neural networks.

$wpxb$

→ McCulloch-Pitts model of a neuron

→ feed-forward network → No loop

→ Recurrent → loop

→ Different types of activation functions

→ taxonomy of feed forward & recurrent

→ feed forward
- → single layer
- → Multi
- → Radial Basis function net

Recurrent → Competitive.
→ Kohonen's SOM
→ Hopfield
→ ART model

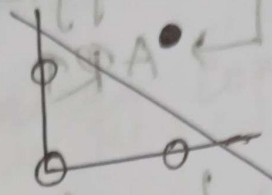
→ Perceptron learning algorithm.
↳ update the weight

→ Back-Propagation algorithm
↳ neural network design (book) chapter 11

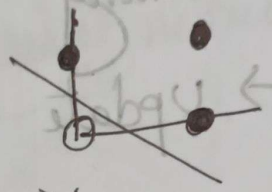
→ Well-known learning algorithm
level ↳ Supervised – Decision tree.
↳ Unsupervised – Centroid cluster
no level ↳ Reinforcement. – Q
↳ critics

→ hidden unit in a 2-dimensional input space

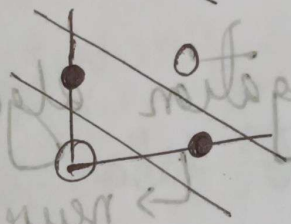
AND



OR



XOR



→ ANN

↳ what?

↳ why powerful?

CT: Sentence → clause

query

inference

Using predicate logic

1.) Marcus was a man

2.) Marcus was a Pompeian

3.) All Pompeians were Romans.

4.) Caesar was a Ruler

5.) All pompeians were either loyal to caesar or hated him

6.) Every one is loyal to someone.

7.) People only try to assassinate rulers they are not loyal to.

8.) Marcus tried to assassinate Caesar

$\Rightarrow$ 1.) Marcus was a man.

man (Marcus).

2.) Marcus was a Pompeian.

Pompeian (Marcus)

3.) All pompeians were Romans.

$\forall x$: Pompeian(x) $\rightarrow$ Roman(x).

4.) Caesar was a Rulere

ruler (caesar)

5.) All pompeians were either loyal to caesar or hated him.

Inclusive-or

$\forall x$: Roman(x) $\rightarrow$ loyalto(x, caesar) $\vee$ hate(x, caesar)

exclusive-or

$\forall x$: Roman(x) $\rightarrow$ (loyalto(x, caesar) $\wedge$ $\neg$ hate(x, caesar)) $\vee$ ($\neg$ loyalto(x, caesar) $\wedge$ hate(x, caesar))

6.) Every one is loyal to someone.

$\forall x : \exists y : \text{loyalto}(x, y)$

7) People only try to assassinate rulers they are not loyal to.

$\forall x : \forall y : \text{Person}(x) \wedge \text{ruler}(y) \wedge \text{tryassassinate}(x, y) \rightarrow \neg \text{loyalto}(x, y)$

8.) Marcus tried to assassinate caesar

tryassassinate (Marcus, caesar)

→ was Marcus loyal to caesar?

man (Marcus)

ruler (caesar)

tryassassinate (marcus, caesar)

$\Downarrow \quad \forall x : \text{man}(x) \rightarrow \text{Person}(x)$

$\neg$ loyalto (Marcus, caesar)

Reasoning

1.) Marcus was a Pompeian

2) All Pompeians died when the volcano erupted in 79 A.D.

3.) It is now 2008 A.D.

Is Marcus Alive?

1.) Marcus was a Pompeian

Pompeian (Marcus)

2) All Pompeians died when the volcano erupted in 79 A.D.

erupted (volcano, 79) $\wedge$ $\forall x$: Pompeian (x) $\rightarrow$ died (x, 79)

3.) It is now 2008 {A.D.}

now = 2008

$\forall x : \forall t_1 : \forall t_2 : died(x, t_1) \wedge greater\text{-}than(t_2, t_1)$

$\rightarrow dead(x, t_2)$

1.) Marcus was a man

man (Marcus)

2.) Marcus was a pompeian

Pompeian (Marcus)

3.) Marcus was born in 40 A.D.

born (Marcus, 40)

4.) All men are mortal.

$\forall x : man(x) \rightarrow mortal(x)$

5.) All pompeians died when the volcano erupted in 79 A.D.

erupted (volcano, 79) $\wedge \forall x : [Pompeian(x)$

$\rightarrow died(x, 79)]$

6.) No mortal lives longer than 150 years.

$\forall x : \forall t_1 : \forall t_2 :$ mortal $(x) \wedge$ born $(x, t_1) \wedge$ gt $(t_2 - t_1, 150) \rightarrow$ dead (x, t_2)

7.) It is now 1991

now $= 1991$

8.) Alive means not dead

$\forall x : \forall t : [\text{alive}(x,t) \rightarrow \neg \text{dead}(x,t)] \wedge$
$[\neg \text{alive}(x,t) \rightarrow \text{dead}(x,t)]$

9.) If someone ~~died~~ dies, then he is dead at all later times.

$\forall x : \forall t_1 : \forall t_2 :$ died $(x, t_1) \wedge$ gt (t_2, t_1)
$\rightarrow$ dead (x, t_2)

Query: Was Marcus alive?

$$\left.\begin{array}{l} born(Marcus, t_1) \\ gt(now - t_1, 150) \end{array}\right\} \rightarrow alive(Marcus, now)$$

A set of facts about Marcus:

1.) man (Marcus)

2.) Pompeian (Marcus)

3.) born (Marcus, 40)

4.) $\forall x$: man(x) $\rightarrow$ mortal (x)

5.) $\forall$: Pompeian(x) $\rightarrow$ died (x, 79)

6.) erupted (volcano, 79)

7.) $\forall x : \forall t_1 : \forall t_2$: mortal (x) $\wedge$ born $(x, t_1) \wedge gt(t_2 - t_1, 150) \rightarrow dead(x, t_2)$

8.) now = 1991

9.) $\forall x : \forall t : [alive(x,t) \rightarrow \neg dead(x,t)]$

$\wedge [dead(x,t) \rightarrow \neg alive(x,t)]$

10. $\forall x : \forall t_1 : \forall t_2 : died(x,t_1) \wedge gt(t_2, t_1)$
$\rightarrow dead(x, t_2)$

One way of proving that Marcus is dead:

$\neg alive(Marcus, now)$
↑ (9, substitution)
$dead(Marcus, now)$
↑ (10, substitution)
$died(Marcus, t_1) \wedge gt(now, t_1)$
↑ (5, substitution)
$Pompeian(Marcus) \wedge gt(now, 79)$
↑ (2)
$gt(now, 79)$
↑ (8, substitute equals)
$gt(1991, 79)$
↑ nil (compute gt)

Another way of proving that Marcus is dead:

¬alive(Marcus, now)

↑ (9, substitution)

dead(Marcus, now)

↑ (7, substitution)

mortal(Marcus) ∧ born(Marcus, t_1) ∧ gt(now − t_1, 150)

↑ (4, substitution)

man(Marcus) ∧ born(Marcus, t_1) ∧ gt(now − t_1, 150)

↑ (1)

born(Marcus, t_1) ∧ gt(now − t_1, 150)

↑ (3)

gt(now − 40, 150)

↑ (8)

gt(1991 − 40, 150)

↑ (compute minus)

gt(1951, 150)

↑ nil compute gt.

Axioms in clause form:

1) man (Marcus)

2) Pompeian (Marcus)

3) born (Marcus, 40)

4) $\neg man(x_1) \vee mortal(x_1)$

5) $\neg pompeian(x_2) \vee died(x_2, 79)$

6) erupted (volcano, 79)

7) $\neg mortal(x_3) \vee \neg born(x_3, t_1) \vee \neg gt(t_2 - t_1, 150) \vee \neg dead(x_3, t_2)$

8) now = 1991

9a) $\neg alive(x_4, t_3) \vee \neg dead(x_4, t_3)$

9b) $dead(x_5, t_4) \vee alive(x_5, t_4)$

10) $\neg died(x_6, t_5) \vee \neg gt(t_6, t_5) \vee dead(x_6, t_6)$.

Prove: $\neg alive(Marcus, now)$

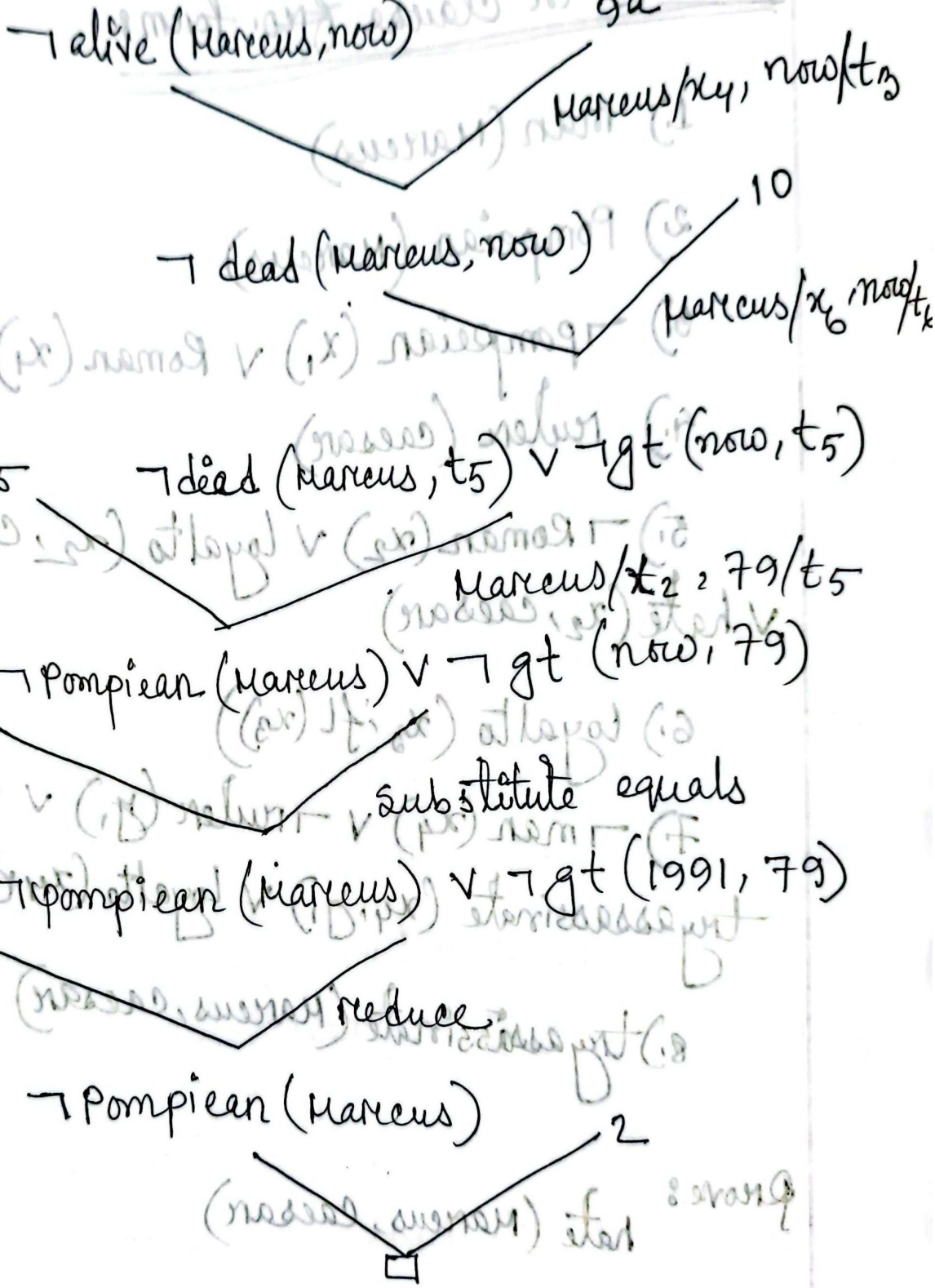


Fig: Using ~~equal~~ Resolution with equality & Reduce

Axioms in clause form:

1.) man (Marcus)

2.) Pompeian (Marcus)

3.) $\neg$ pompeian (x_1) $\vee$ Roman (x_1)

4.) ruler (caesar)

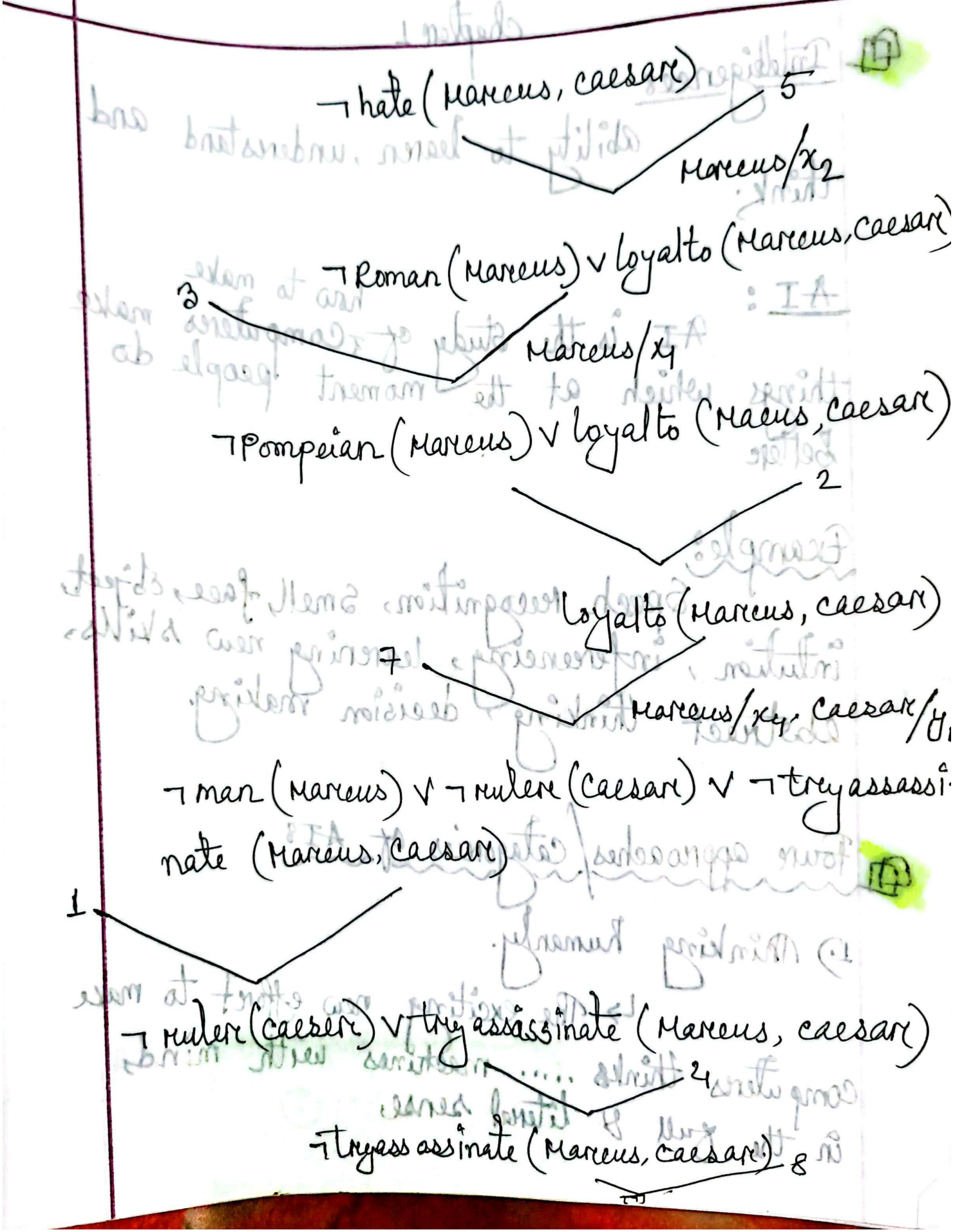
5.) $\neg$ Roman (x_2) $\vee$ loyalto (x_2, caesar) $\vee$ hate (x_2, caesar)

6.) loyalto (x_3, fl (x_3))

7.) $\neg$ man (x_4) $\vee$ $\neg$ ruler (y_1) $\vee$ $\neg$ tryassassinate (x_4, y_1) $\vee$ loyalto (x_4, y_1)

8.) tryassassinate (Marcus, caesar)

Prove: hate (Marcus, caesar)


¬hate(Marcus, caesar)
5
Marcus/x2
¬Roman(Marcus) ∨ loyalto(Marcus, caesar)
3
Marcus/x4
¬Pompeian(Marcus) ∨ loyalto(Maeus, caesar)
2
loyalto(Marcus, caesar)
7
Marcus/x4, caesar/
¬man(Marcus) ∨ ¬ruler(caesar) ∨ ¬tryassassi-
nate(Marcus, caesar)
1
¬ruler(caesar) ∨ tryassassinate(Marcus, caesar)
4
¬tryassassinate(Marcus, caesar)
8

chapter 1

Intelligence: ability to learn, understand and think.

AI: AI is the study of how to make computers make things which at the moment people do better.

Example: Speech recognition, Smell, face, object, intuition, inferencing, learning new skills, abstract thinking, decision making.

Four approaches/catagories of AI:

1) Thinking Humanly.

↳ The exciting new effort to make computers thinks ..... machines with minds in the full & literal sense.

2) Thinking rationally.

↳ The study of computations that make it possible to perceive, reason, and act.

3.) Acting humanly.

↳ The study of how to make computers do things at which, at the moment, people do better.

4.) Acting rationally:

↳ AI is concerned with intelligent behaviour in artifacts.

Acting humanly: The turing test:

→ The computers would need to posses the following capabilities:

① Natural language processing: To

enable it to communicate successfully in English.

2) Knowledge Representation : To store what knows & hears.

3.) automated reasoning : To use the stored information to answer questions & to draw new conclusions.

4.) Machine learning : To adapt to new circumstances and to detect & extrapolate patterns.

To pass the total turing test, the computer will need -

① Computere visions - To perceive objects
② Robotics - To manipulate objects

Thinking humanly : (The cognitive modeling approach)
→ through (introspection ①

→ through psychological experiments.
→ brain imaging.

Thinking Rationally: The "Laws of thoughts" approach

Acting : The Rational agent approach

Task domains of AI:

1) Mundane tasks:
   → perception:
      → Vision
      → Speech
   → Natural language:
      → Understanding
      → generation
      → Translation
   → Common sense reasoning.
   → Robot control

2) Formal Tasks:

→ Games : chess, checkers etc

→ Mathematics : Geometry, logic, proving properties of programs.

3.) Expert tasks:

→ Engineering

→ Scientific analysis.

→ Medical diagnosis

→ Financial analysis

AI Techniques:

→ Intelligence requires knowledge

→ Knowledge possess less desireable properties such as

→ voluminous

→ hard to characterize accurately

→ constantly changing.

→ differs from data that can be used

AI technique is a method that exploits knowledge that should be represented in such a way that -

→ Knowledge captures generalization.

→ It can be understood by people who must provide it.

→ It can be modified to correct errors.

→ It can be used in variety of situations.

The state of the art:

1. Computer beats human in a chess game
2. Computer - human conversation using speech recognition.
3. Expert systems control a spacecraft.
4. Robot can walk on stairs and hold a cup of water.

5. Language translation for webpages.

6. Home appliances use fuzzy logic.

chapter : 02

problem solving = Searching for a goal state

Water jug problem:

1) State: (x,y)

x = 0, 1, 2, 3 or 4 ; y = 0, 1, 2, 3

2) Start State:

(0,0)

3) Goal state:

(2,n) for any n

4) Attempting to end up in a goal state.

State Space:

1) (x, y) → (4, y)
   If x < 4

2) (x, y) → (x, 3)
   If y < 3

3) (x, y) → (x − d, y)
   If x > 0

4) (x, y) → (x, y − d)
   If y > 0

5) (x, y) → (0, y)
   If x > 0

6) (x, y) → (x, 0)
   If y > 0

7) (x, y) → (4, y − (4 − x))
   If x + y ≥ 4, y > 0

8.) $(x,y) \rightarrow (x-(3-y), 3)$

If $x+y \geq 3, x>0$

9.) $(x,y) \rightarrow (x+y, 0)$

If $x+y \leq 4, y>0$

10.) $(x,y) \rightarrow (0, x+y)$

If $x+y \leq 3, x>0$

11.) $(0,2) \rightarrow (2,0)$

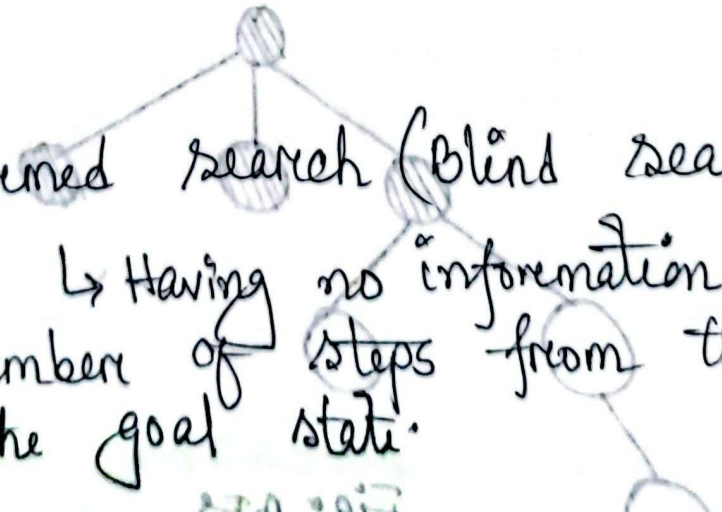
12) $(2,y) \rightarrow (0,y)$

(0,0)	(0,0)	(0,0)
(0,3)	(0,3)	(0,3)
(3,0)	(3,0)	(3,0)
(3,3)	(3,3)	(3,3)
(4,2)	(4,2)	(4,2)
(0,2)	(0,2)	(0,2)
(2,0)	(2,0)	(2,0)

Search Strategies:

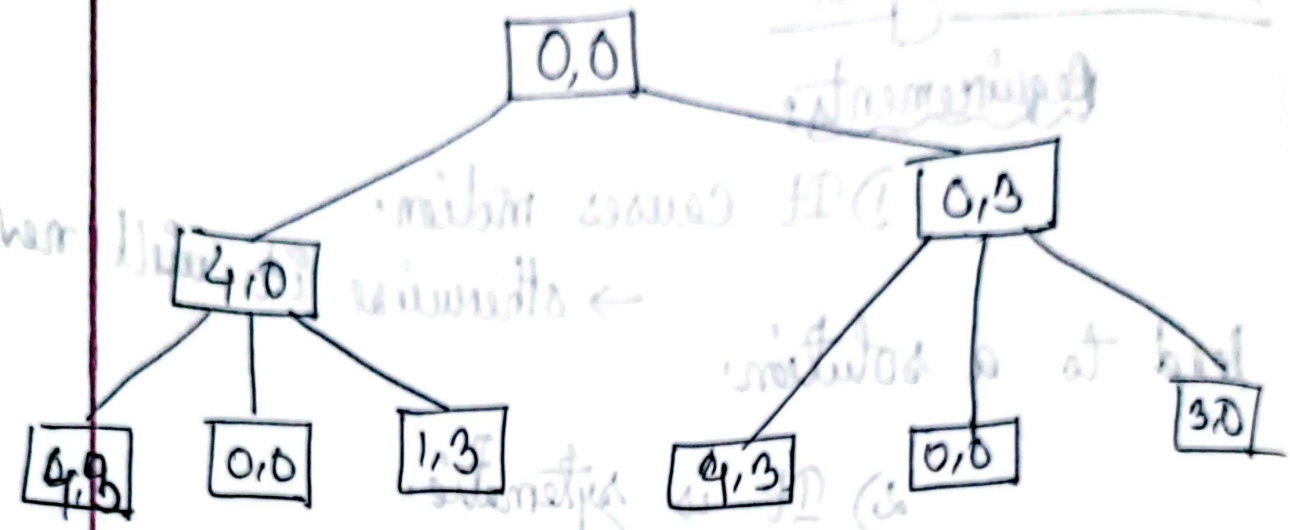
Requirements:

1) It causes motion.
   → otherwise it will never lead to a solution.
2) It is systematic.
   → otherwise it may use more steps than necessary.
3) It is efficient.
   → Find a good, not necessarily the best solution.



→ Uninformed search (Blind search)
   ↳ Having no information about the no. of number of steps from the current state to the goal state.

→ Informed search (Heuristic search)
   ↳ more effecient than uninformed search



Blind Search:

↳ BFS: Expand all the nodes of one level first.

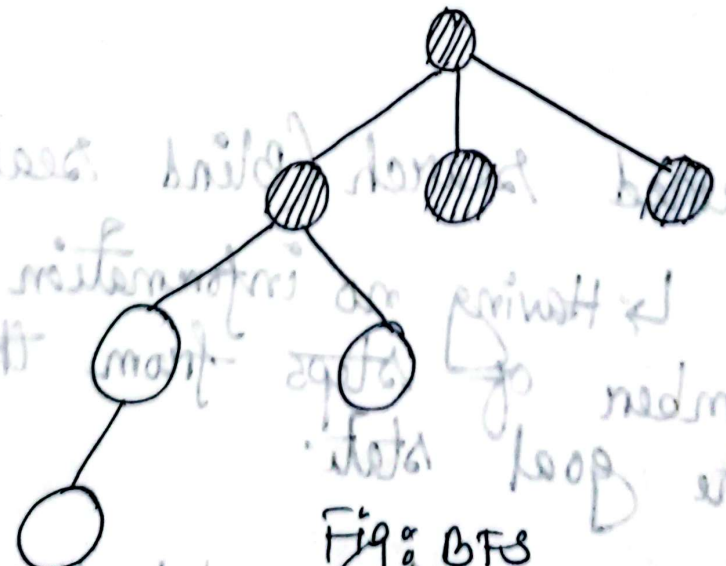


Fig: BFS

→ DFS: Expand one of the nodes at the deepest level.

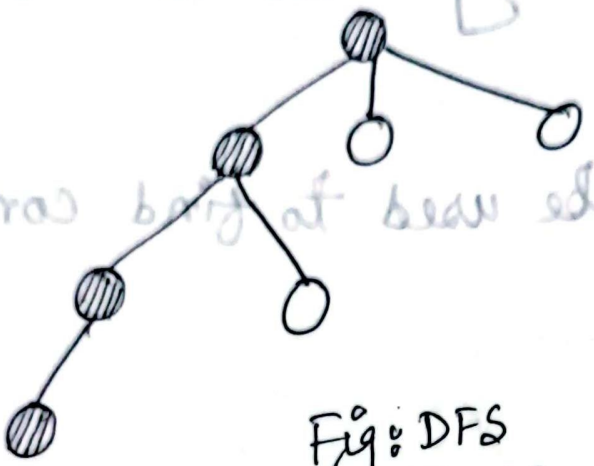


Fig: DFS

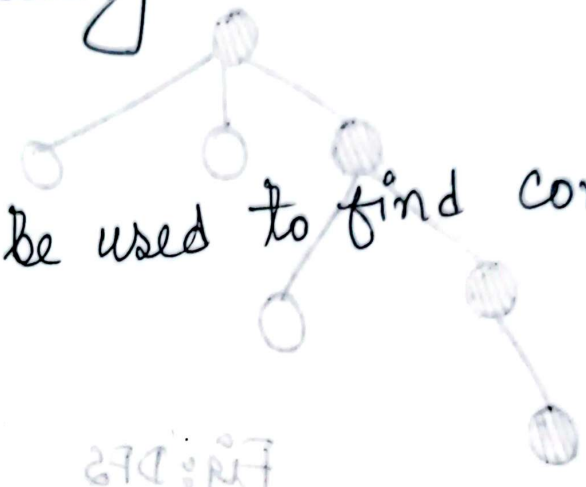
Advantages of BFS:

1) Completeness: BFS is guaranteed to find the shortest path because it explores nodes level by level.

2) Optimality: BFS guarantees an optimal solution in terms of the number of edges traversed.

3.) Memory efficiency:

BFS often requires less memory, especially when the branching factor is small.



4.) Can be used to find connected components.

Disadvantages of BFS:

1.) Memory consumption:

BFS can be memory intensive in graphs with high branching factors.

2.) Inefficiency for deep trees:

In very deep trees, BFS might need to explore a large number of nodes before finding a solution, which can be inefficient.

3.) BFS may be slower for dense graphs.

Advantages of DFS:

1.) DFS is often easier to implement recursively.

2.) uses less memory than BFS, especially when dealing with deep trees.

3.) DFS can find multiple solutions.

Disadvantages of DFS:

1.) Completeness:

DFS is not guaranteed to the find the shortest path.

2.) Non-optimality:

DFS doesn't guarantee an optimal solution.

3.) Susceptible to infinite loops.

Problem characteristics:

1) Is the problem decomposable?
2) Can solution steps be ignored/undone?
3) Is the universe predictable?
4) Is a good solution absolute/relative?
5) Is the solution a state or a path?
6) What is the role of knowledge?
7) Does the task require human-interacting?

Chapter 3

→ Generate-and-test

→ Hill climbing

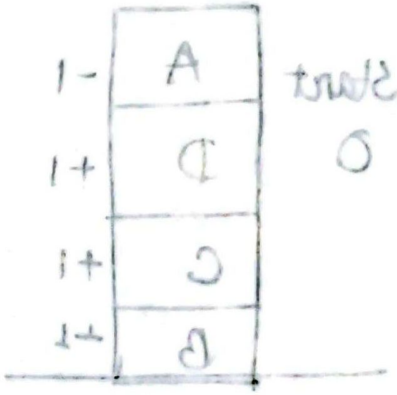
Searching for a goal state = climbing to the top of a hill

Generate-and-test + direction to move

Heuristic function:

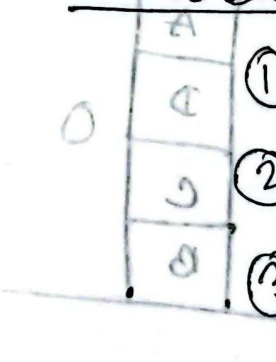
↳ to estimate how close a given state is to a goal state.

→ Simple Hill climbing

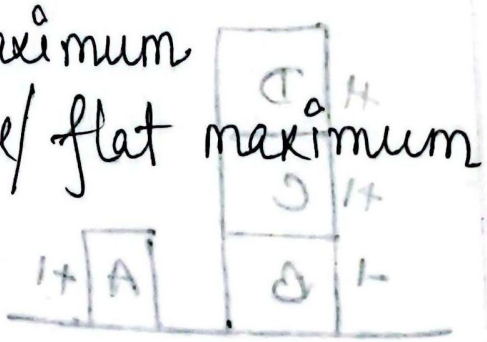


Simple Hill climbing:

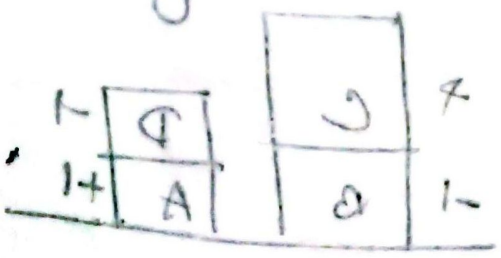
Drawbacks



① local maximum
② Plateaue/ flat maximum
③ Ridge



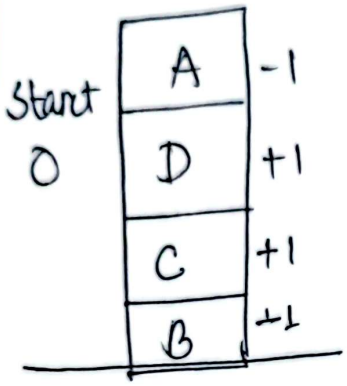
→ local search algorithm
→ Greedy
→ No backtracking

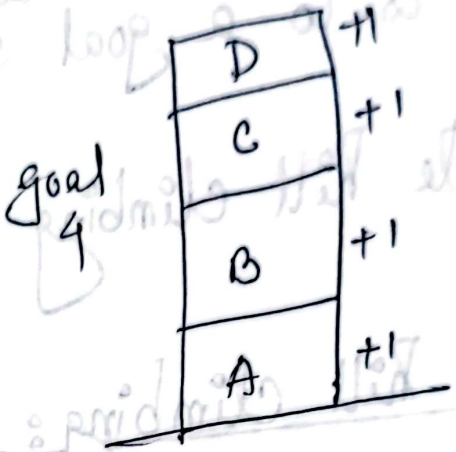


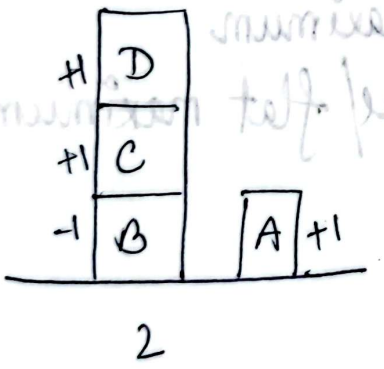
Steepest - Ascent Hill climbing

↳ gradient search

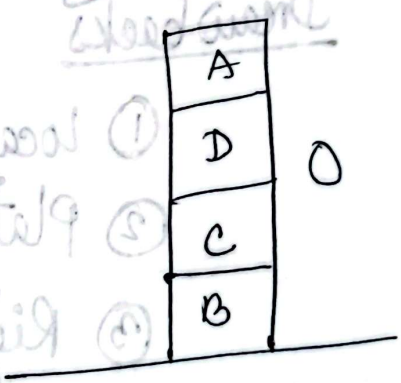
→ Local Heuristic:

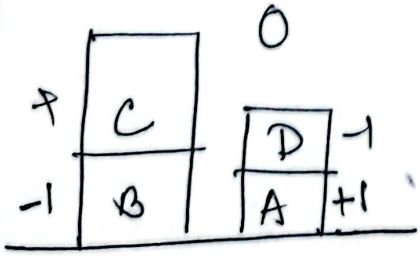


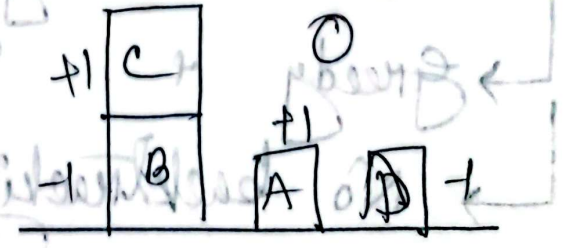




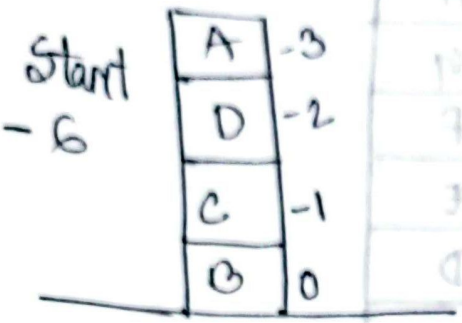
2

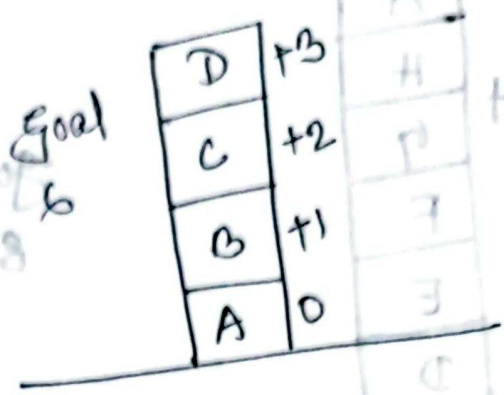


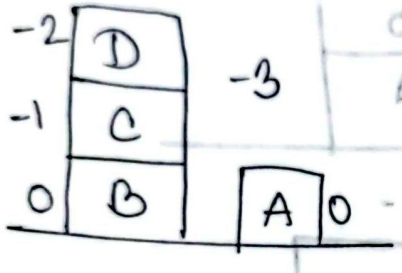


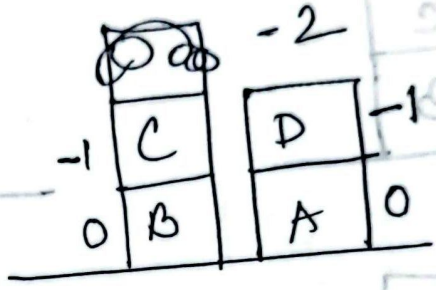


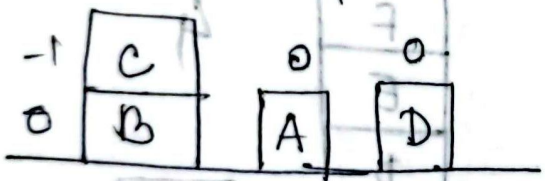
Global Heuristic search

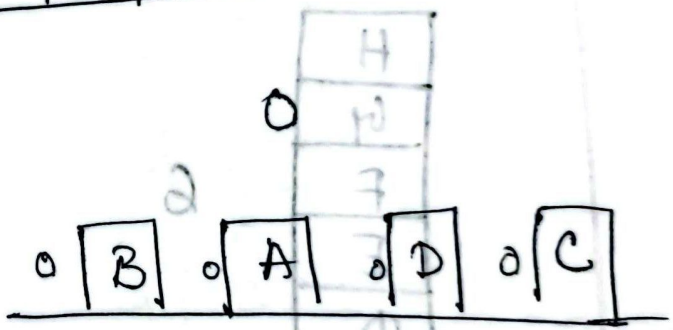


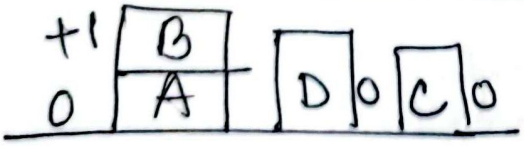


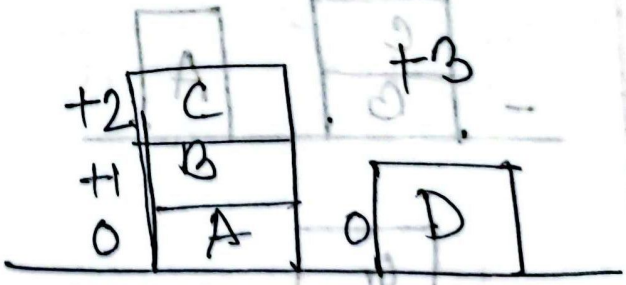


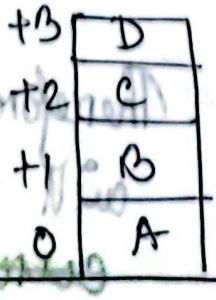




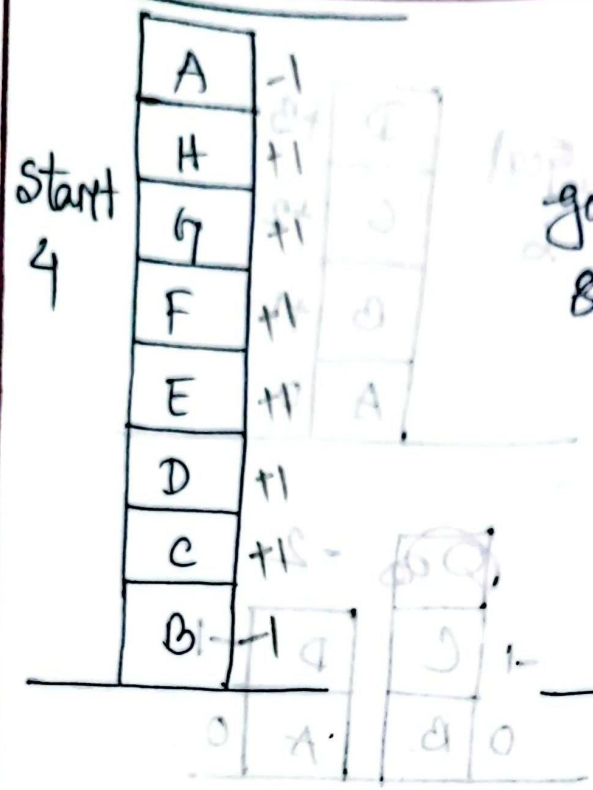


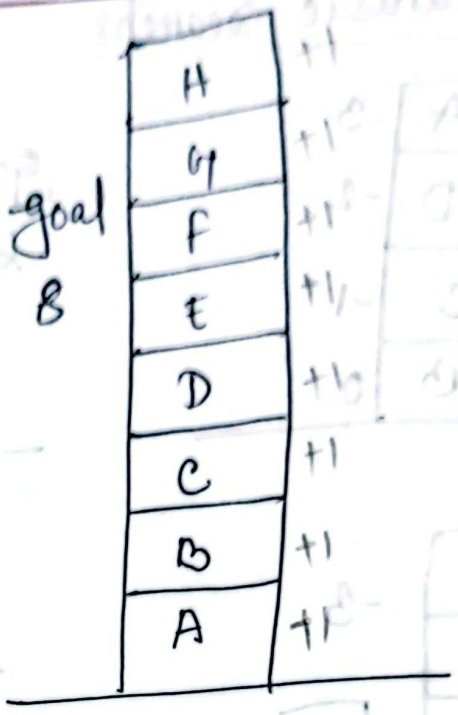


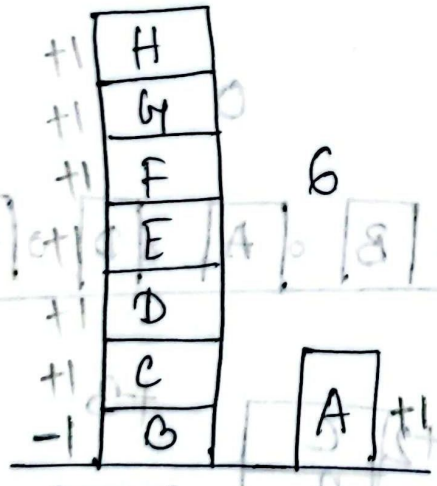


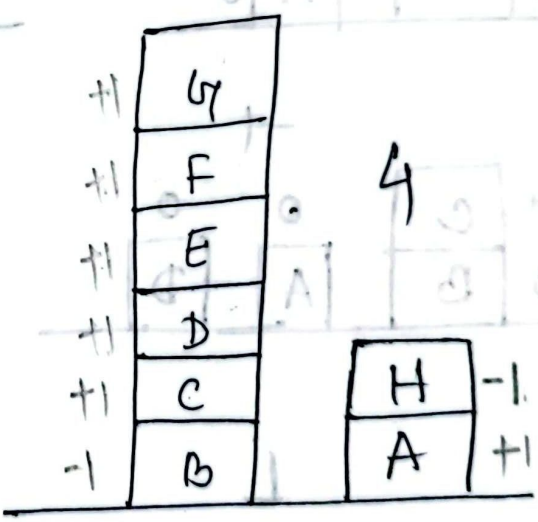


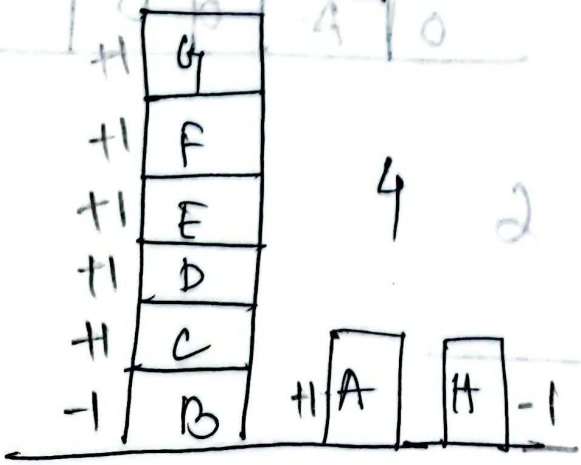
Local Heuristics





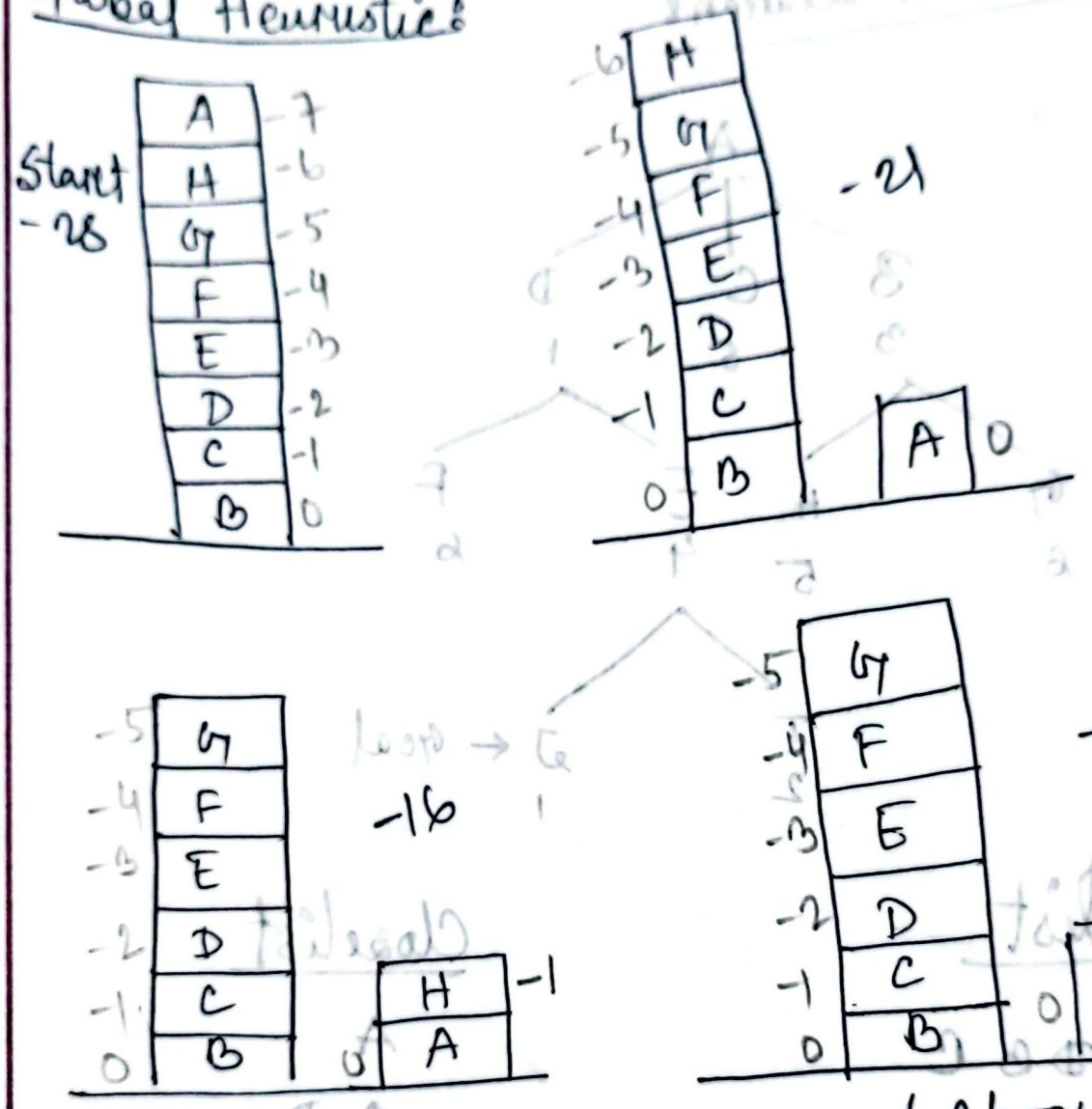






Therefore, Hill climbing will Reject as these current states have lower scores than the current state.

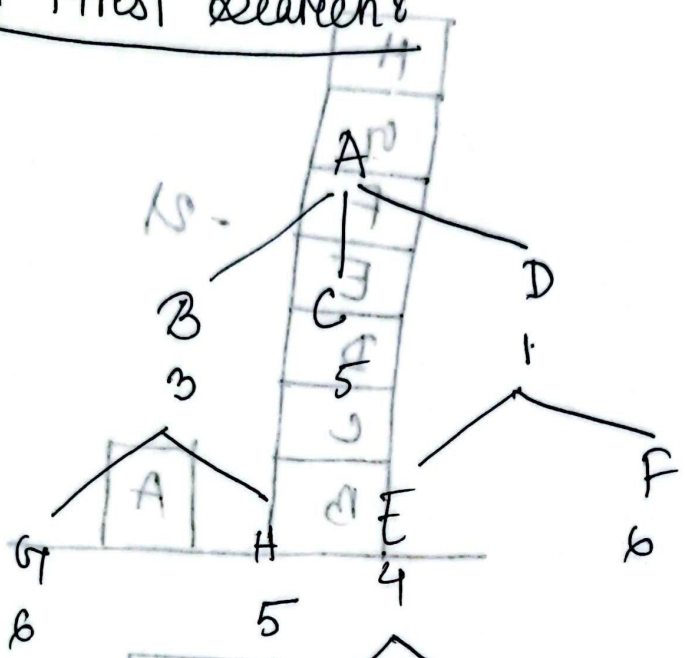
Global Heuristic:

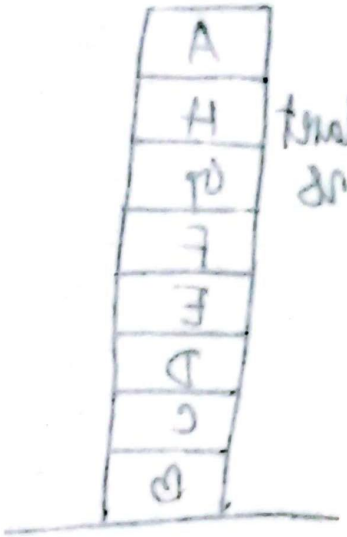


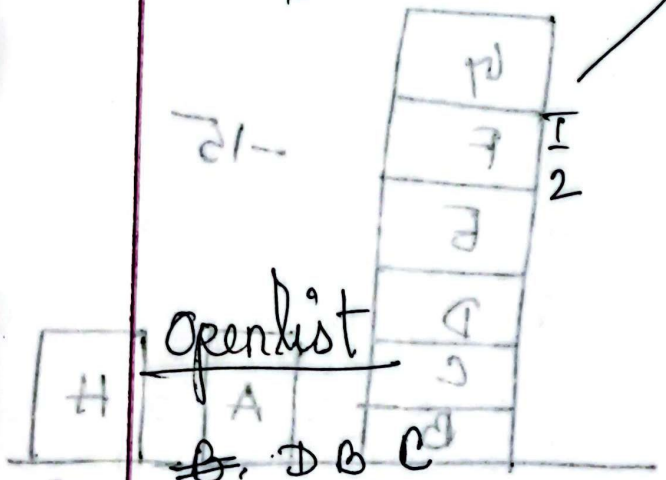
Simulated annealing:

↳ Backtracking

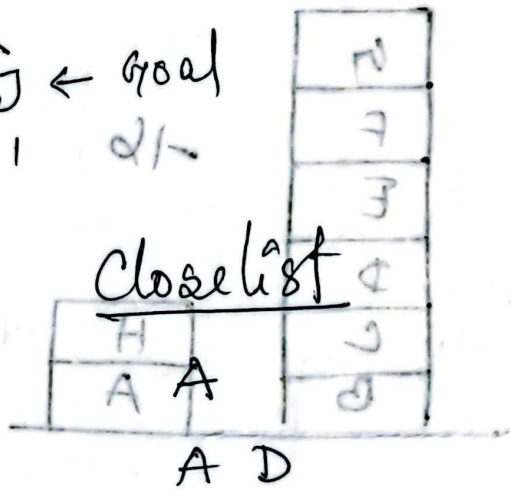
Best-First Search:







Openlist



Closelist

B E C F

E C H F G

J I C H F G

A D B

A D B E

A D B E J

A* Algorithm :

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

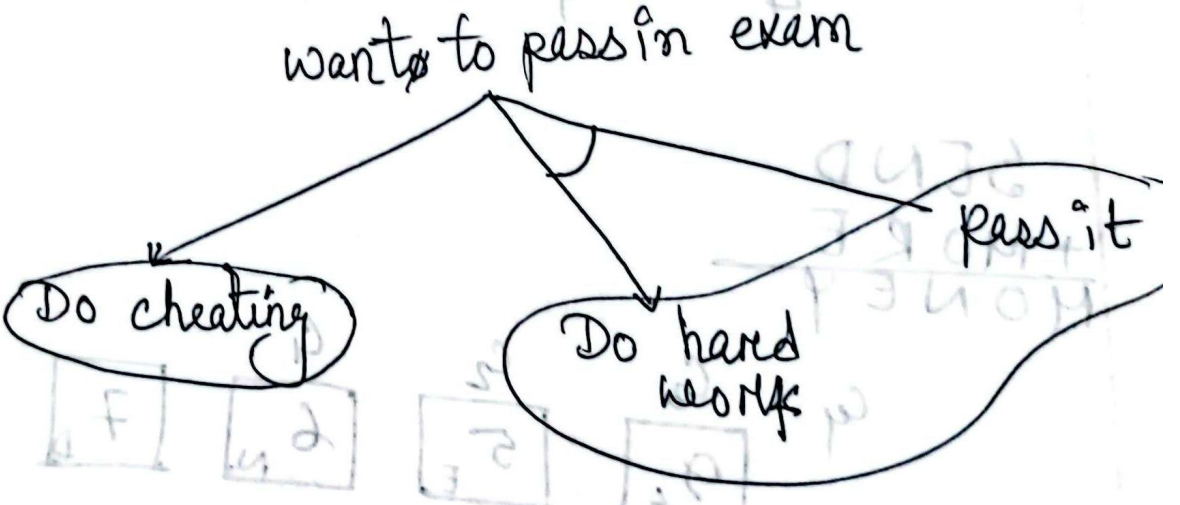
$$f^*(n) = g^*(n) + h^*(n)$$

$g^*(n)$ → depth factore

$h^*(n)$ → heuristic factore

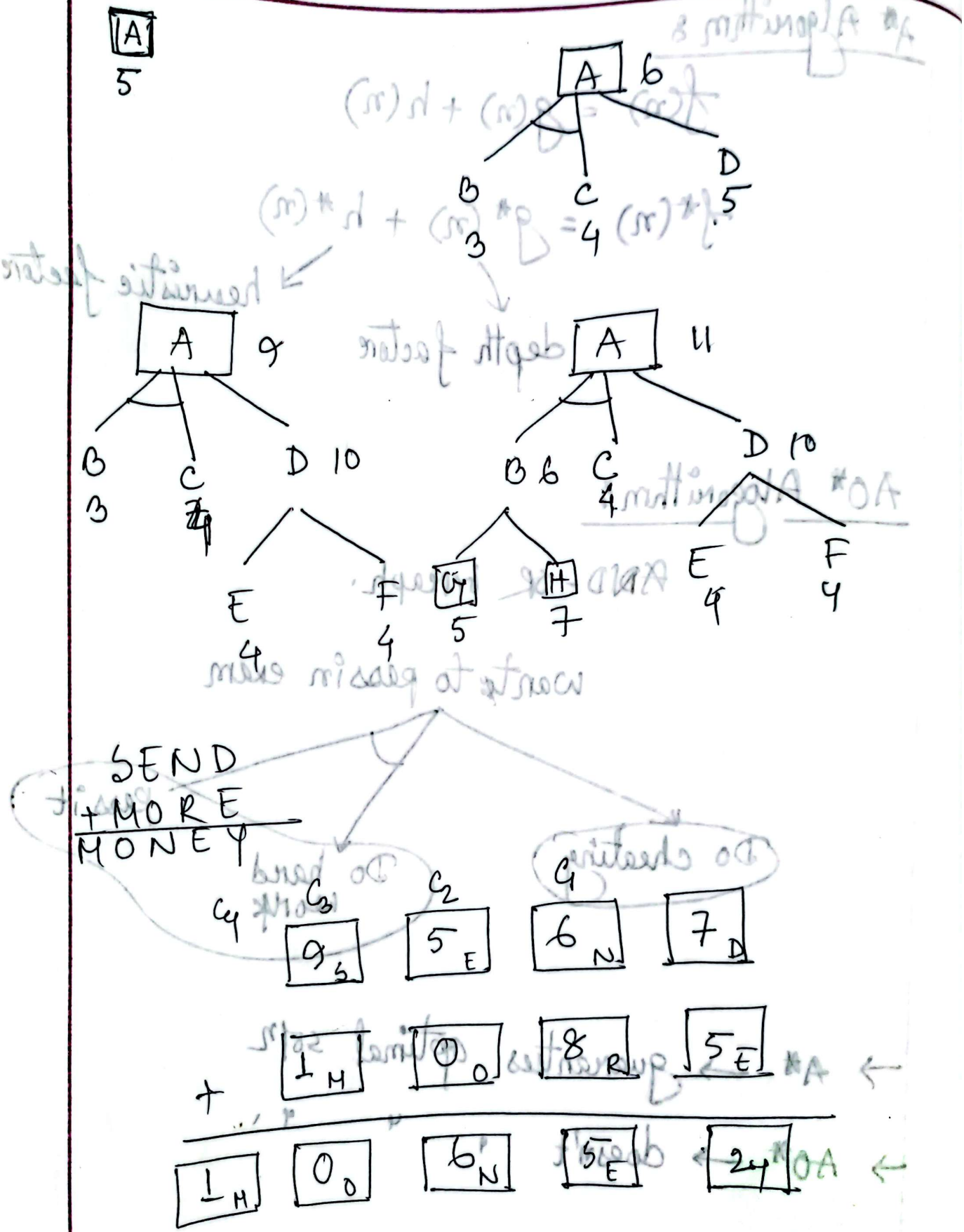
AO* Algorithm :

AND-OR breaph.



→ A* → guarantees optimal soln

→ AO* → doesn't


A
5
A
6
B
3
C
4
D
5
A
9
B
3
C
4
D
10
E
4
F
4
A
11
B
6
C
4
D
10
G
5
H
7
E
4
F
4
SEND
+MORE
MONEY
9 S
5 E
6 N
7 D
1 M
0 O
8 R
5 E
1 M
0 O
6 N
5 E
2

$S + M \geq 10$

$S + 1 \geq 10$

$\Rightarrow S = 9$

$E + 0 = N$ if $C_2 = 0$ ✗

↳ 0 ↳ $C_2 = 1$

let, $E = 5$

$\Rightarrow E + 0 + C_2 = N$

$\Rightarrow N = 6$

$N + R = E$

$\Rightarrow 6 + R = 5$

~~$\Rightarrow 6 + 9 =$~~

if $C_1 = 1$

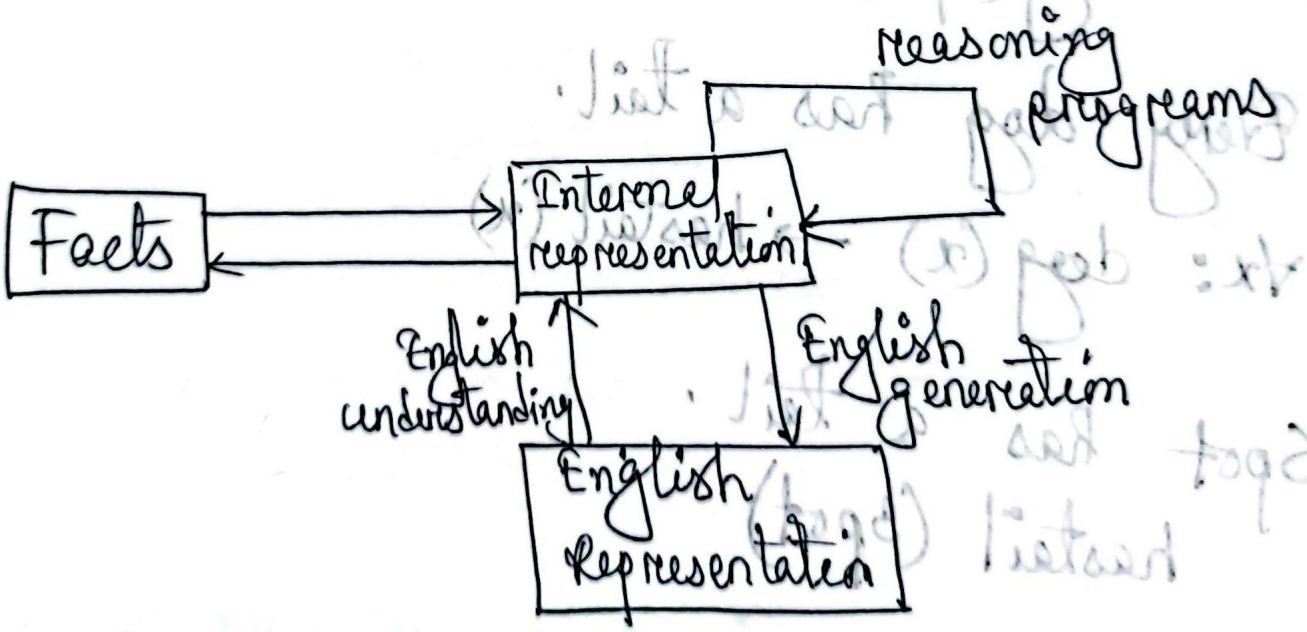
$6 + 8 + 1 = 5$

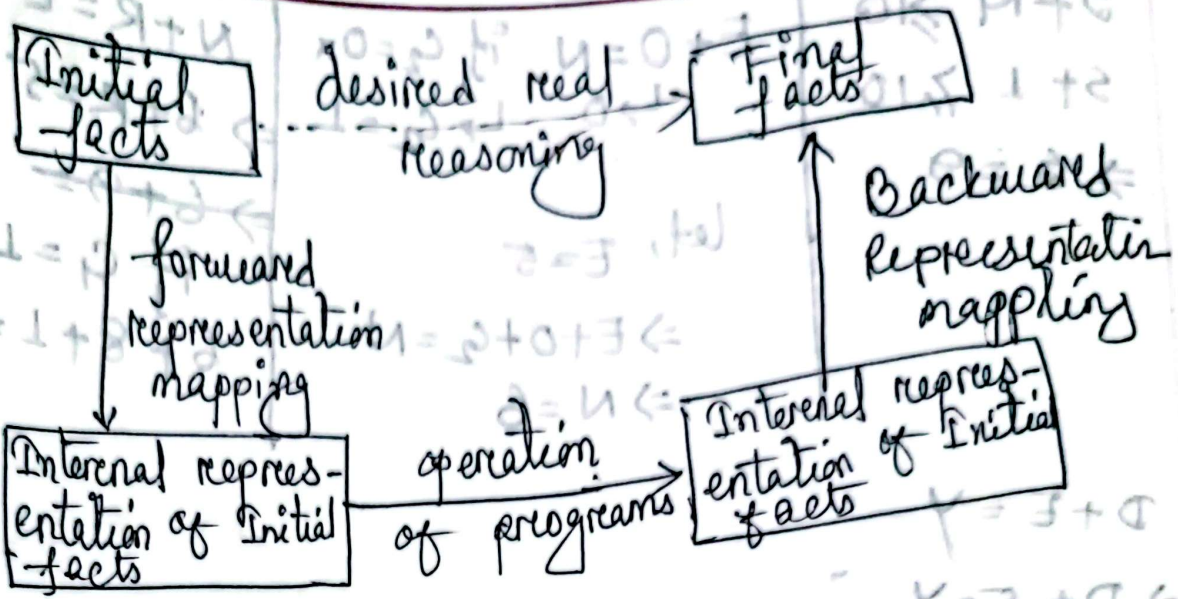
$D + E = Y$

$\Rightarrow D + 5 = Y$

$\Rightarrow 7 + 5 = 2$

D: 5 ✗, 6 ✗, 7 ✓, 8 ✗, 9 ✗





Spot is a dog.

dog(Spot)

Every dog has a tail.

$\forall x$: dog(x) $\rightarrow$ hastail(x)

Spot has a tail.

hastail(Spot)

Approaches to KR:

1. Simple relational Knowledge
2. Inheritable
3. Inferential
4. procedural
5. Choosing the granularity.

Chapter-5

Symbols

$\neg$ = negation

$\vee$ = Disjunction/or

$\wedge$ = Conjunction/and

$\rightarrow$ = if then

$\leftrightarrow$ = iff

using propositional logic:

→ Theorem proving is decidable

→ Can not represent objects & quantification.

Using predicate logic:

→ Theorem proving is semi-decidable.

→ Can represent objects & quantification.

1 Marcus was a man.

man (Marcus)

2 Marcus was a pompeian.

pompeian (Marcus)

3 All pompeians were Romans.

$\forall x$: pompeian (x) → Roman (x)

4 Caesar was a ruler.

ruler (caesar)

5 All pompeians were either loyal to caesar or hated him.

$\forall x$: Roman (x) → loyalto (x, caesar) ∨ hate (x, caesar)

6 Everyone is loyal to someone

$\forall x : \exists y : \text{loyalto}(x, y)$

7 People only try to assassinate rulers they are not loyal to.

$\forall x : \forall y : \text{person}(x) \wedge \text{ruler}(y) \wedge \text{tryassassinate}(x, y) \rightarrow \neg \text{loyalto}(x, y)$

8 Marcus tried to assassinate caesar

tryassassinate (Marcus, Caesar)

9 All men are people

$\forall x : \text{man}(x) \rightarrow \text{person}(x)$

Was Marcus loyal to caesar?

Nill

↓(1)

man (Marcus)

↓(9)

Person (Marcus)

↓(8)

Person (Marcus) ∧ tryassassinate (Marcus, Caesar)

↓(4)

Person (Marcus) ∧ tryassassinate (Marcus, Caesar)
∧ ruler (Caesar)

↓(7)

¬ loyalto (Marcus, Caesar)

□ was Marcus hates Caesar?

¬loyalto(Marcus, caesar)

↓(2)

Pompeian(Marcus) → ¬loyalto(Marcus, caesar)

↓(3)

Roman(Marcus) → ¬loyalto(Marcus, caesar)

↓(5)

hate(Marcus, caesar)

1.) Marcus was man

man(Marcus)

2) Marcus was a pompeian

Pompeian(Marcus)

3.) Marcus was born in 40A.D.

born(Marcus, 40)

(4.) All men are mortal.

$\forall x : man(x) \rightarrow mortal(x)$

5.) All pompeians died when the volcano erupted in 79 A.D.

erupted (volcano, 79) $\wedge$ $\forall x$: [pompeian(x) $\rightarrow$ died (x, 79)]

6.) No mortal lives longer than 150 years.

$\forall x : \forall t_1 : \forall t_2 :$ mortal(x) $\wedge$ born(x, t_1) $\wedge$ gt($t_2 - t_1$, 150) $\rightarrow$ dead(x, t_2)

7.) It is now 1991.

now = 1991

8.) Alive means not dead.

$\forall x : \forall t :$ [alive(x, t) $\rightarrow$ $\neg$ dead(x, t)] $\wedge$ [$\neg$ dead(x, t) $\rightarrow$ alive(x, t)]

9.) If someone dies, then he is dead at all later times.

$\forall x : \forall t_1 : \forall t_2 :$ died(x, t_1) $\wedge$ gt(t_2, t_1) $\rightarrow$ dead(x, t_2)

Axioms in clause form:

1.) man (Marcus)

2.) Pompeian (Marcus)

3.) $\neg$ pompeian (x_1) $\vee$ Roman (x_1)

4.) ruler (caesar)

5.) $\neg$ Roman (x_2) $\vee$ loyalto (x_2, caesar) $\vee$ hate (x_2, caesar)

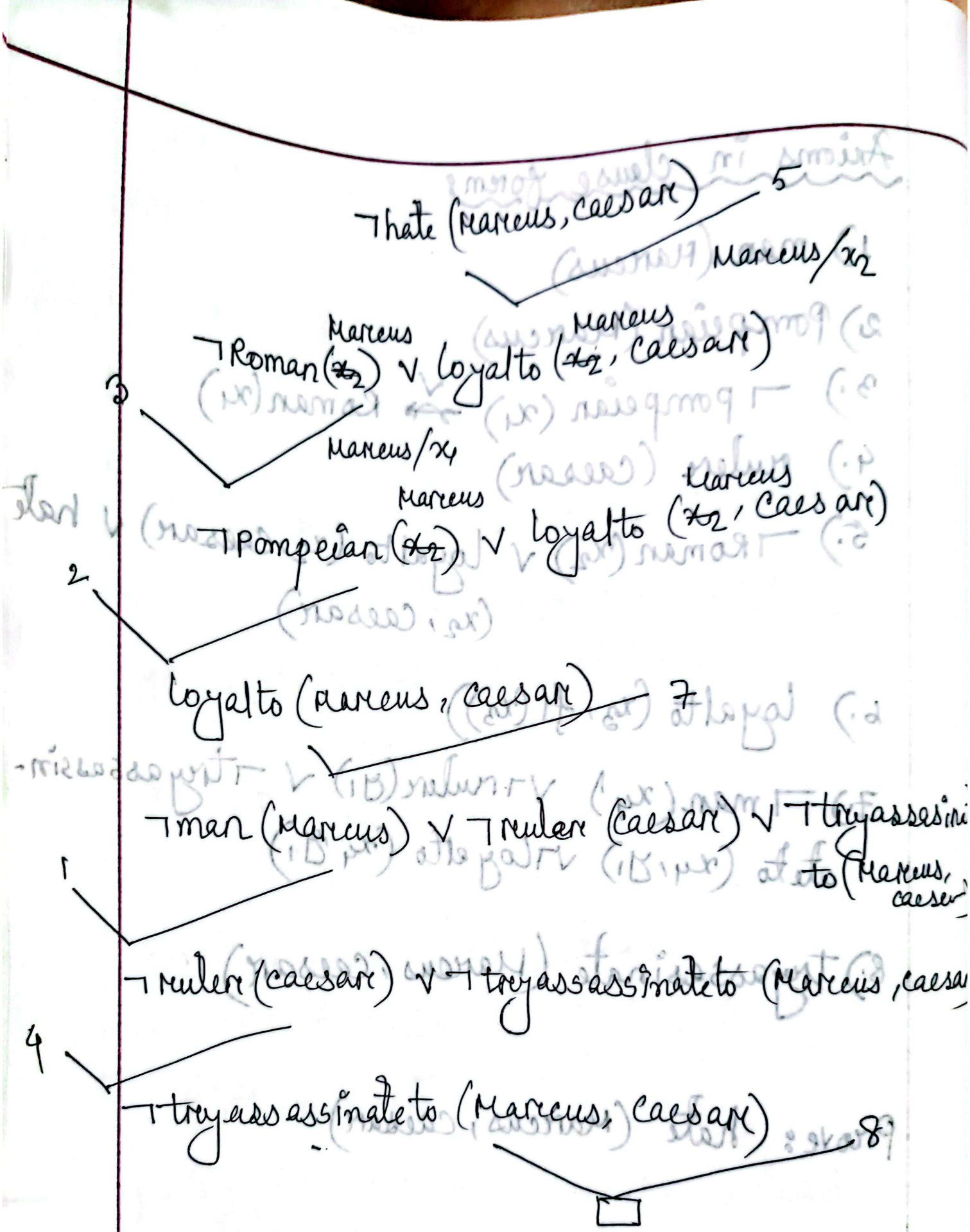
6.) loyalto (x_3, f1 (x_3))

7.) $\neg$ man (x_4) $\vee \neg$ ruler (y_1) $\vee \neg$ tryassassin ateto (x_4, y_1) $\vee \neg$ loyalto (x_4, y_1)

8.) tryassasinate (Marcus, caesar)

Prove: hate (Marcus, caesar).

□


¬hate (marcus, caesar)
5
marcus/x2
¬Roman(x2) ∨ loyalto (x2, caesar)
marcus
marcus
3
marcus/x4
¬Pompeian(x2) ∨ loyalto (x2, caesar)
marcus
marcus
2
loyalto (marcus, caesar)
7
¬man (marcus) ∨ ¬ruler (caesar) ∨ ¬tryassesin
to (marcus,
caesar
1
¬ruler (caesar) ∨ ¬tryassassinateto (marcus, caesa
4
¬tryassassinateto (marcus, caesar)
8

Axioms in clause form:

1.) man (Marcus)

2.) Pompeian (Marcus)

3.) born (Marcus, 40)

4.) $\neg man(x_1) \vee mortal(x_1)$

5.) $\neg pompeian(x_2) \vee died(x_2, 79)$

6.) erupted (volcano, 79)

7.) $\neg mortal(x_3) \vee \neg born(x_3, t_1) \vee \neg gt(t_2 - t_1, 150) \vee dead(x_3, t_2)$

8.) now = 2008

9a.) $\neg alive(x_4, t_3) \vee \neg dead(x_4, t_3)$

9b.) $dead(x_5, t_4) \vee alive(x_5, t_4)$

10.) $\neg died(x_6, t_5) \vee \neg gt(t_6, t_5) \vee dead(x_5, t_6)$

Prove: $\neg alive(Marcus, now)$

alive(Marcus, now) — 9a

¬dead(Marcus, now) — 10

5 — ¬dead(Marcus, t_5) ∨ ¬gt(now, t_5)

¬pompeian(Marcus) ∨ ¬gt(now, 79)

Substitute equals

¬pompeian(Marcus) ∨ ¬gt(2008, 79)

reduce

¬pompeian(Marcus)

2 — ☐

alive(Marcus, now) — 9a

¬dead(Marcus, now) — 10

5 ¬dead(Marcus, t_5) ∨ ¬gt(now, t_5)

¬pompeian(Marcus) ∨ ¬gt(now, 79)

Substitute equals

¬pompeian(Marcus) ∨ ¬gt(2008, 79)

reduce

¬pompeian(Marcus) — 2

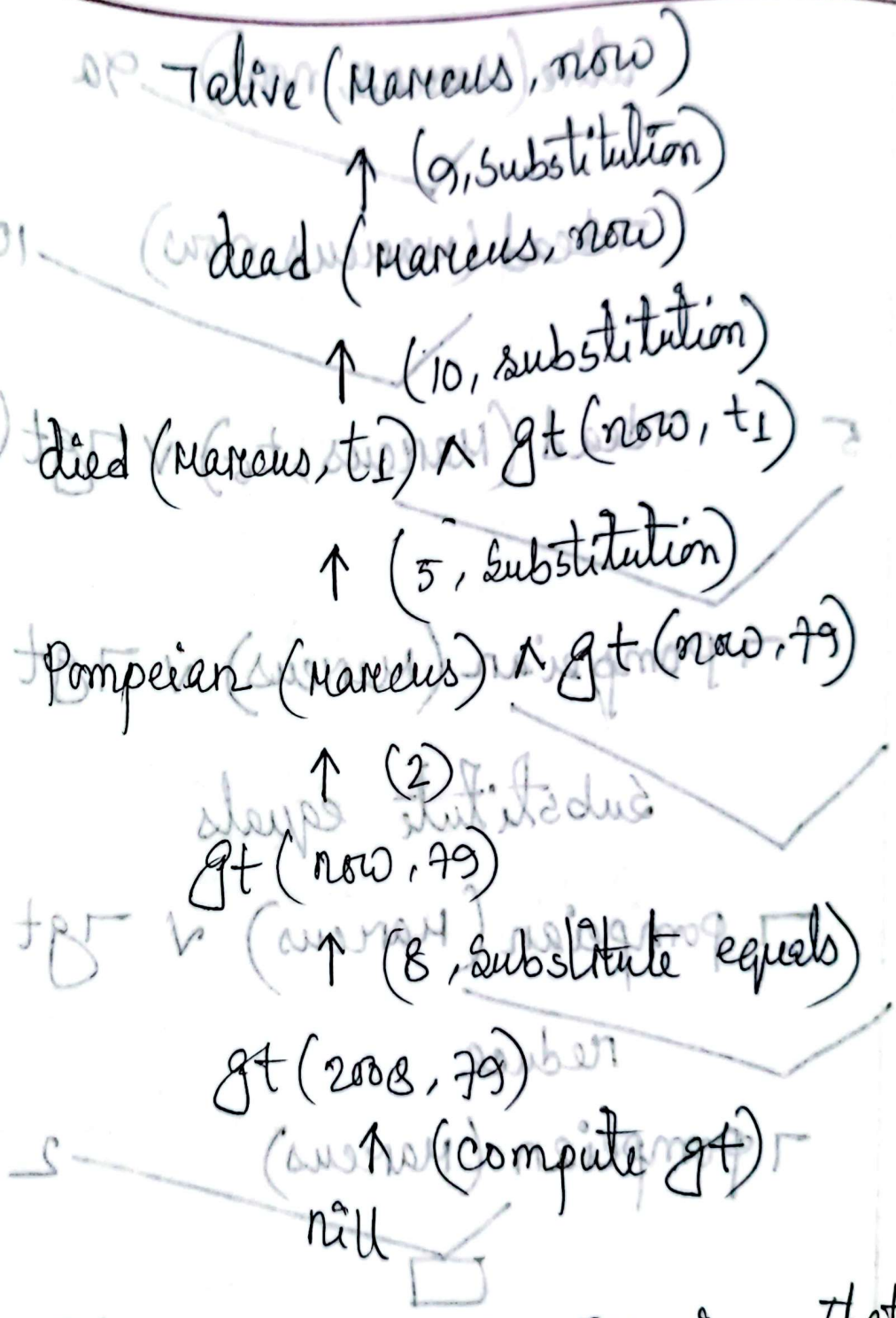


Fig: one way of proving that Marcus is dead

¬alive(Marcus, now)

↑ (9, substitution)

dead(Marcus, now)

↑ (7, substitution)

¬mortal(Marcus) ∧ born(Marcus, t_1) ∧

gt(now − t_1, 150)

↑ (4 substitution)

man(Marcus) ∧ born(Marcus, t_1) ∧

gt(now − t_1, 150)

↑ (1)

born(Marcus, t_1) ∧ gt(now − t_1, 150)

↑ (3)

gt(now − 40, 150)

↑ (8)

gt(2008 − 40, 150)

↑ (Compute gt)

gt(1951, 150)

↑ nil.

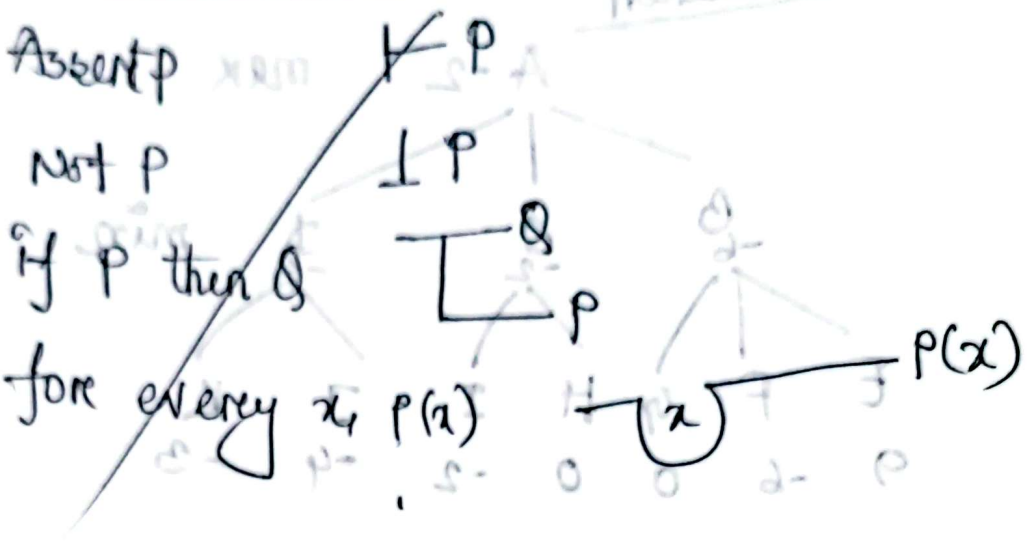
Forward chaining/reasoning:

↳ Starts with the known facts and asserts new facts.

Backward Chaining/reasoning:

↳ Starts with goals and works backward to determine what facts must be asserted so that the goals can be achieved.

Backward chaining	Forward chaining
↓	↓
Answers	Facts
↓	↓
Finding & figuring out the lead	using rules to uncover new info.



Procedural knowledge

↳ Describes how to solve problem

↳ Imperative knowledge

↳ procedure to do something

↳ embedded

Declarative knowledge

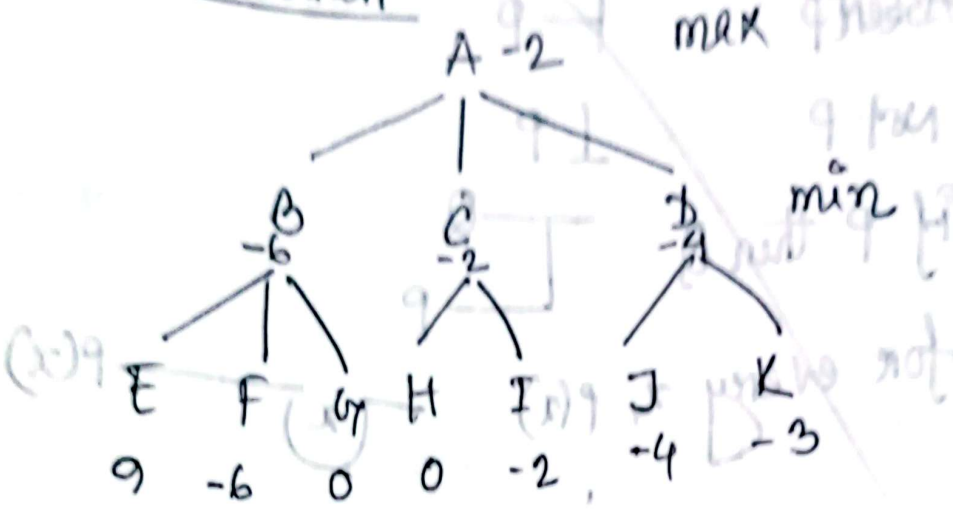
↳ Describes what is known about problem

↳ Descriptive knowledge

↳ Tells us facts.

↳ not embedded

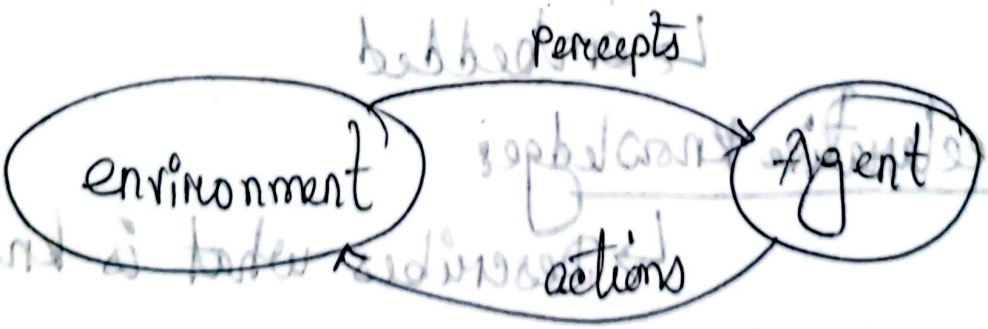
Minimax Search



Two ply search

Agent:

↳ an entity that perceives & acts.



agent = architecture + program

P → Performance measure
E → Environment
A → Actuators
S → Sensors.

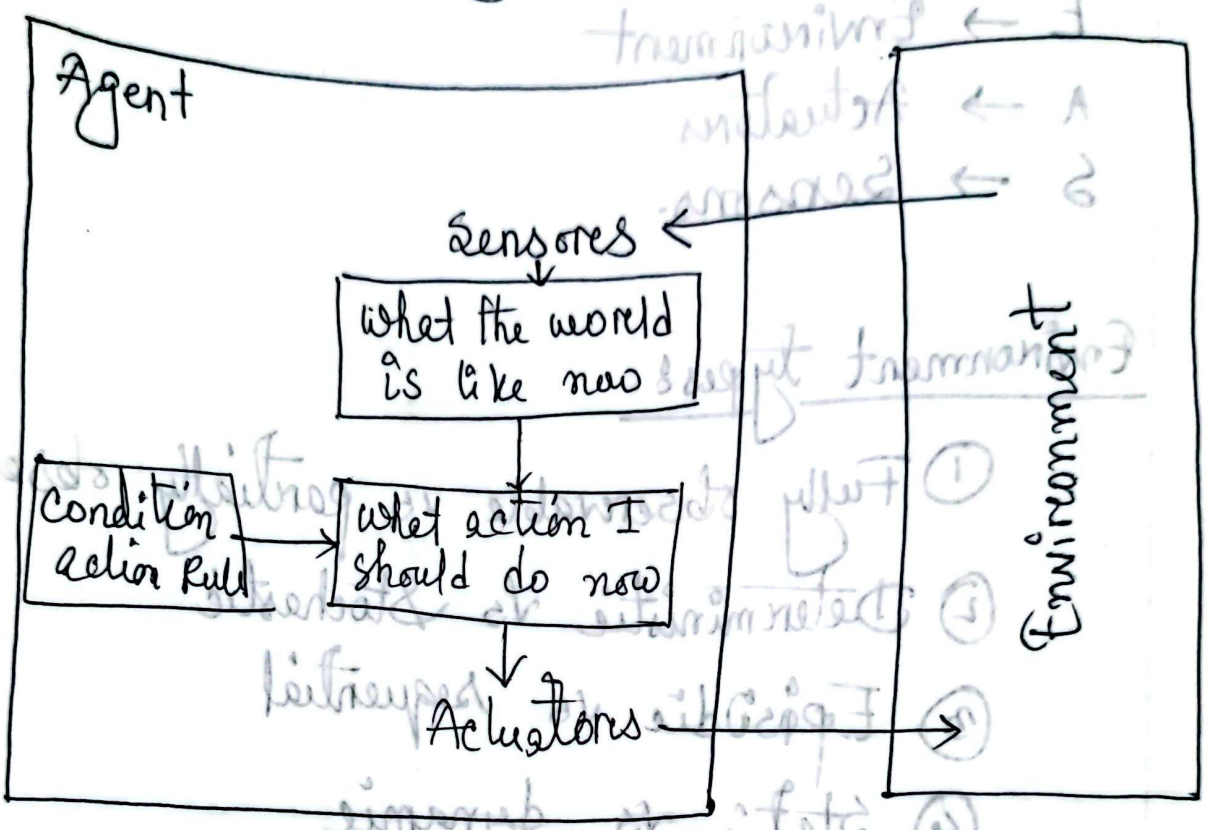
Environment types:

1. Fully observable vs partially observable
2. Deterministic vs Stochastic
3. Episodic vs sequential
4. Static vs dynamic
5. Discrete vs continuous
6. Single agent vs multiagent.

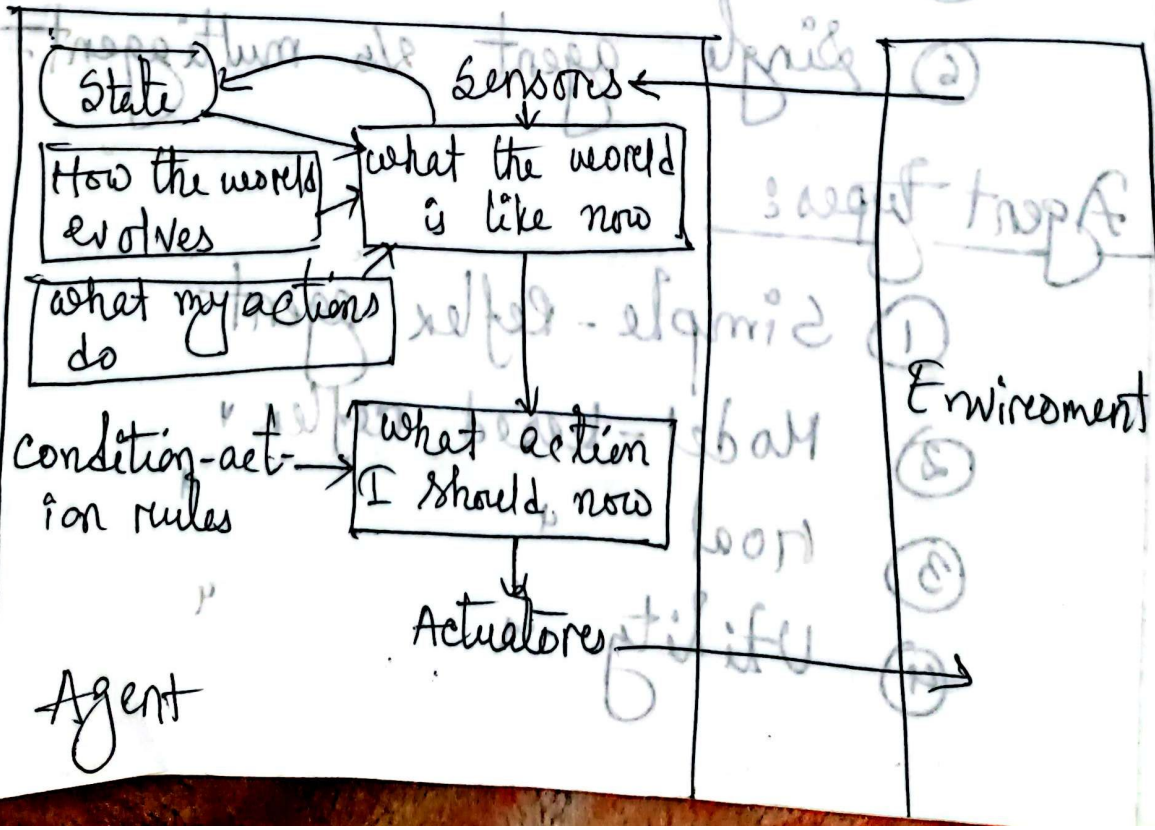
Agent types:

1. Simple-Reflex agent
2. Model-based reflex
3. Goal
4. Utility

Simple Reflex Agent:



Model-based Reflex agent:



Problem type

Deterministic, fully observable → single-state problem

Non-observable → Sensorless problem (conformant problem)

Non-deterministic or partially observable → contingency problem

Unknown state space → exploration problem

A problem is defined by four items:

- Initial state
- actions or successor function
- goal test
- path cost

Uninformed search Strategies:

- BFS
- DFS
- Uniform-cost search

BFS

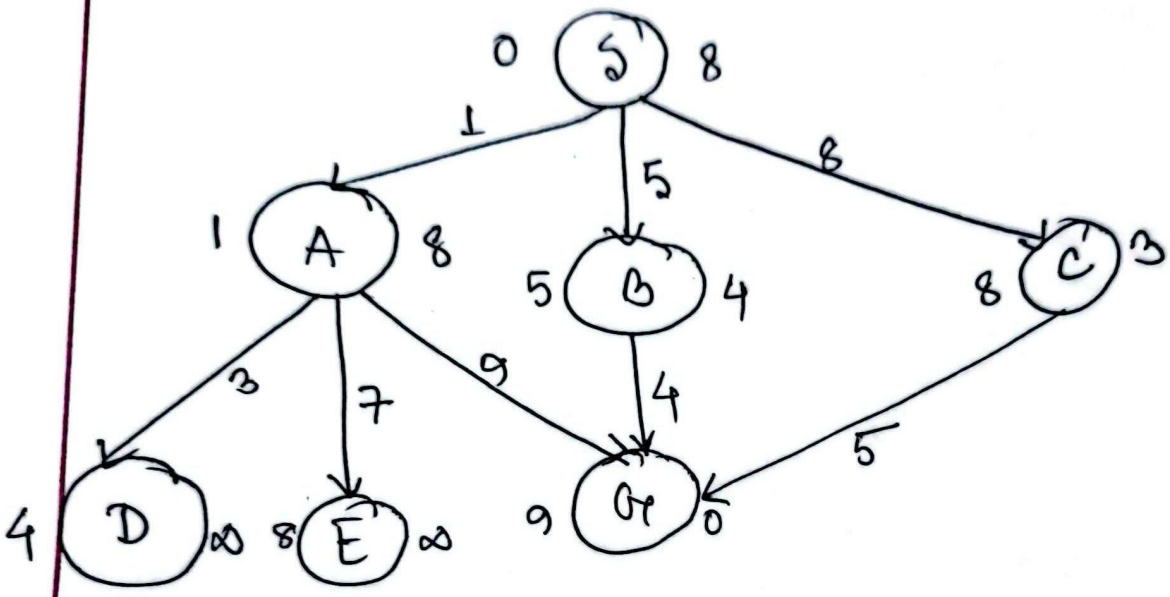
- Time = $O(b^{d+1})$
- Space = 4

DFS

- Time = $O(b^m)$
- Space = $O(bm)$

DFS

Root → left → right



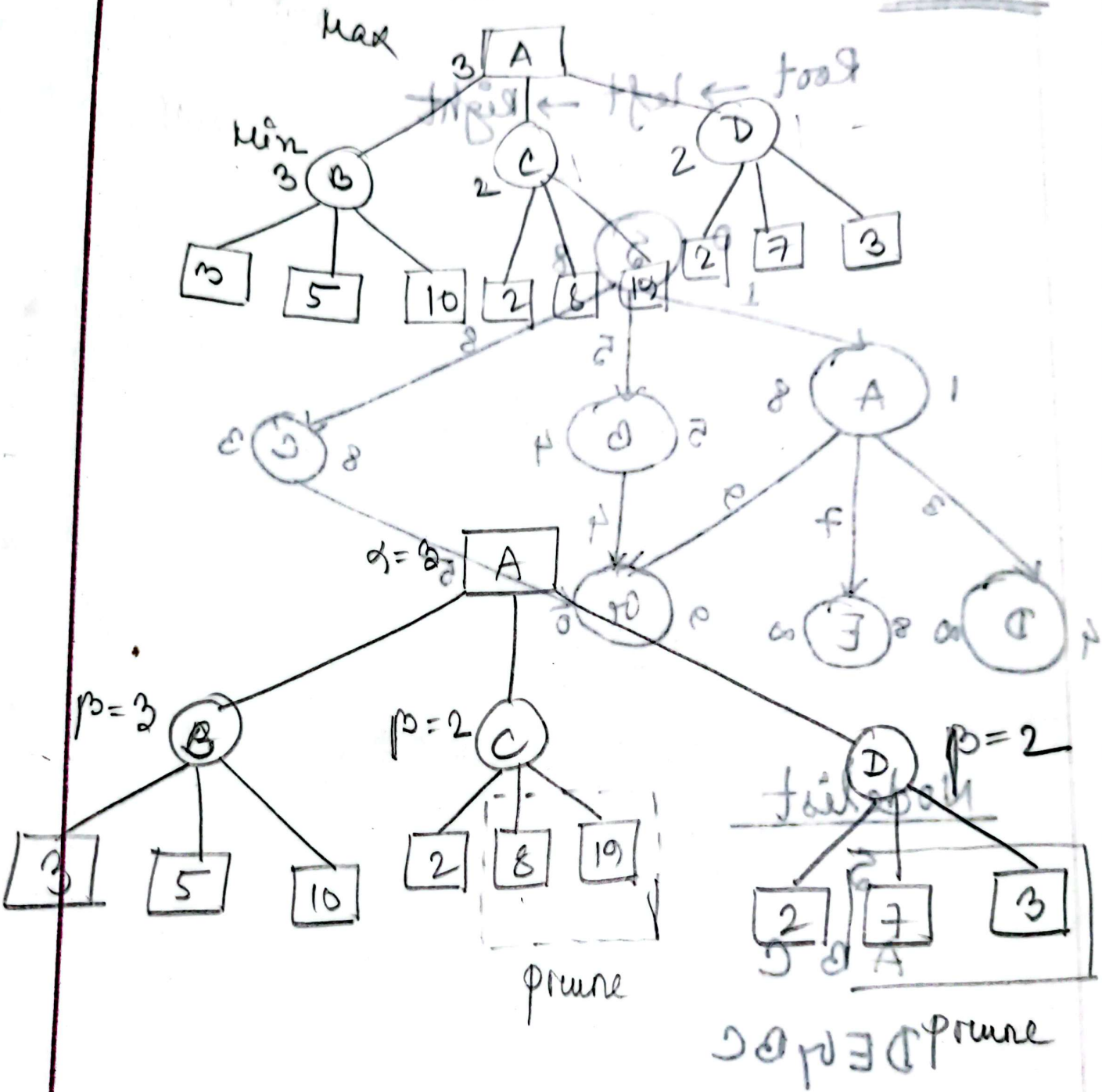
Nodelist

S

A B C

D E G B C

S → A → D → E → G


Max
Min
A
B
C
D
3
2
2
3
5
10
2
8
19
2
7
3
α=3
β=3
β=2
β=2
prune
prune

~~Minimax algorithm~~

Propositional logic :

¬ – not
∧ – and
∨ – or
→ – implies

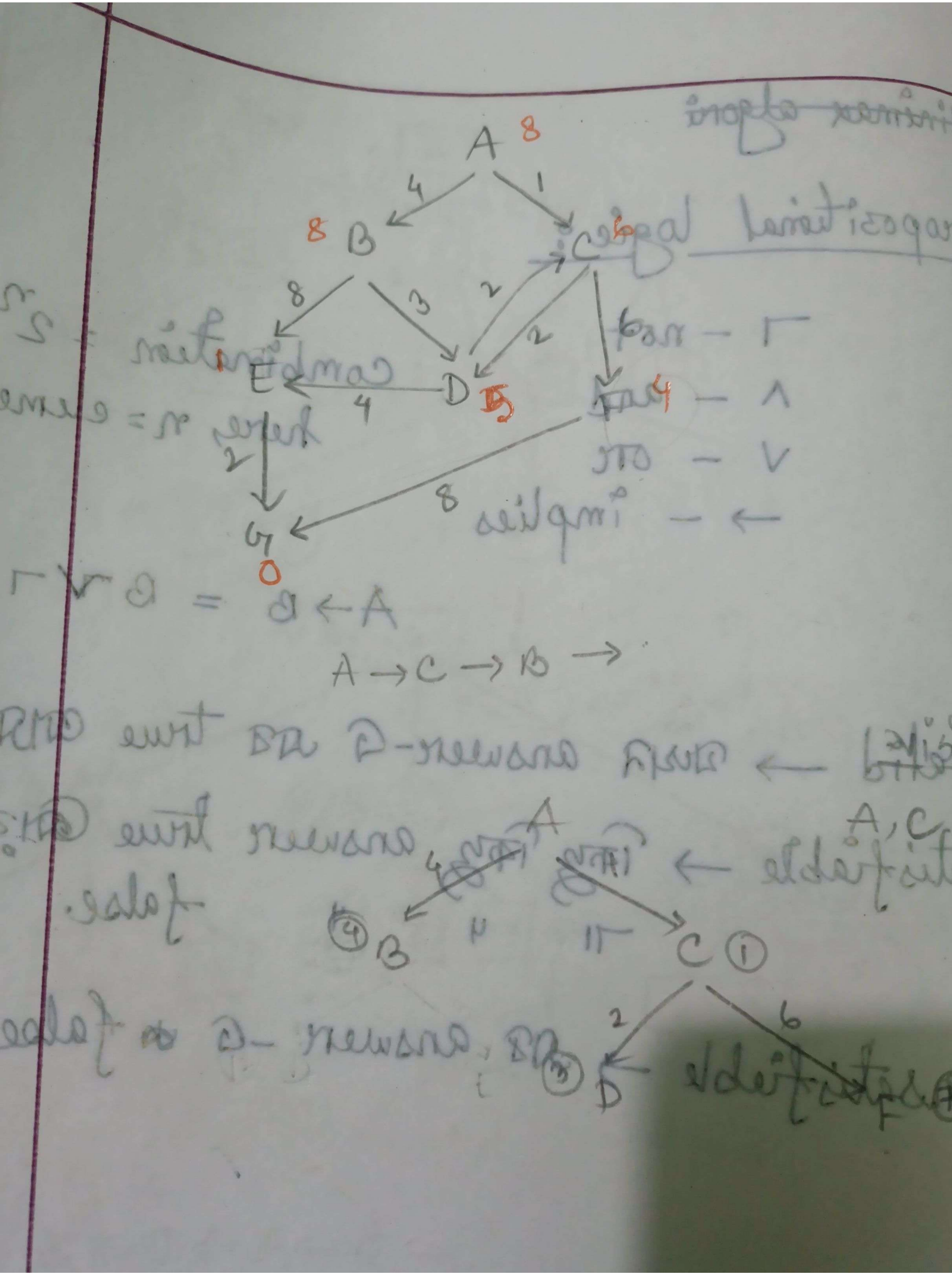
combination $= 2^n$
here, n = element

$A \rightarrow B = B \vee \neg A$

valid → যখন answer-টা সব true আসবে,

Satisfiable → কিছু কিছু answer true আর কিছু false

unsatisfiable → সব answer-টা false


A
B
C
D
E
G
4
1
8
3
2
2
2
4
8
A → C → B →