Name: Kaushiksh Ghosh
Class: D15-C
Roll No: 51

04/05

AI-DS 1

Assignment 1

Q1.] What is AI? Considering the COVID-19 pandemic situation, how AI helped to survive and renovated our way of life with different applications?

→ Artificial Intelligence (AI) refers to the development of systems that can perform tasks requiring human-like intelligence, such as learning, reasoning and problem solving. During the COVID-19 pandemic, AI played a transformative role in healthcare, logistics, and communication. It accelerated drug discovery by analyzing vast datasets, predicted the spread of the virus using predictive models and enabled faster diagnostics through image recognition.

These applications not only helped society survive the crisis but also reshaped how we live, work, and interact with technology.

Q2.] What are AI agents terminology, explain with examples

→ AI agents are autonomous entities that perceive their environment through sensors and act upon it using actuators to achieve specific goals. They can be categorized based on their complexity and functionality. Examples include
- Simple Reflex Agents: React to current percepts (eg. a thermostat adjusting temperature.

· Model based agents: Maintain an internal state to handle partially observable environments (eg self driving cars)

· Goal based agents: Take actions to achieve objective (a delivery robot

· Utility based agents: Maximize performance or utility (eg a stock trading bot optimizing profits).

Q3.] How AI technique is used to solve 8 puzzle problem?

→ The 8 puzzle problem is solved using AI algorithms like A^* or Breadth First Search (BFS). The puzzle is represented as a state space, where each state is a configuration of the tiles.

Start (S)

1	4	3
2	0	5
8	6	7

Goal State (G)

1	2	3
4	5	6
7	8	0

Heuristic function such as the Manhattan Distance guide the search by estimating the cost to reach the goal. These techniques ensure efficient exploration of the state space, avoiding unnecessary paths and reducing computation time.

Q4.] What is PEAS descriptor? Gives PEAS descriptor for following: Taxi Driver, Medical Diagnosis System, A music composer, An aircraft autolander, an essay evaluator, a robotic

Sentry gun for the Keck lab.

→ PEAS stands for Performance, Environment, Actuators and sensors. It defines the framework for designing intelligent agents.

	Performance	Environment	Actuators	Sensors
• Taxi Driver	Safe, fast cost effective	Roads, Traffic passengers, weather	Steering, accelerator brake, display	GPS, cameras, speedometer, fuelgauge
• Medical Diagnosis System	Accurate and timely diagnosis	Patient data, symptoms, doctors	Display results reccommend treatments	Inputs from tests, patient records, and sensors
• Music Composer	Create harmonious and original music	Music Theory, Composition, Software	MIDI controller audio output	Keyboard, mousic, mixer
• Aircraft Autolander	Safe and controlled landing	Aircraft system, weather, traffic control	Engine, Landing gear	Radar, GPS, Air data sensor
• Essay Evaluator	Accurate assessment of essay quality	Essay text Grading criteria	Display screen, printer, communication interface	Text input device, Natural Language.
• Robotic Sentry Gun for Keck Lab.	Effective detection & deterence of intruder	Keck lab premises, intruder behaviour	Gun turret, Movement systems alarm system	Camera, motion, detections infrared sensors

Q5.) Categorize a shopping bot for an offline bookstore according to each of the six dimensions.

→ • Fully / Partially Observable : Partially observable (cannot see all inventory or customer preferences at once.)
• Deterministic / Stochastic : Stocastic. (Customer behaviour and inventory changes are ~~tt~~ unpredicted).
• Episodic / Sequential : Sequential (interactions build over time, e.g., tracking user preferences).
• Static / Dynamic : Dynamic (inventory and prices change over time).
• Discrete / Continuous : Discrete (finite set of actions,
• Single / Multi agent : Single agent (interacts with users independently)

Q6.) Differentiate Model based and Utility based agent.

→ Model-Based Agents : These agents maintain an internal model of the world to handle partially observable environments. They use this model to predict outcomes and make decisions (e.g. a self driving car navigation traffic.)

Utility Based Agents : These agents aim to maximize a utility function which evaluates the desirability of outcomes. They are used in scenarios where trade offs are necessary (e.g. a stock trading bot optimizing profits while minimizing risk).

Q7.] Explain the architecture of a knowledge based agent and learning agent.

→ Knowledge Based Agent:

Architecture:

A knowledge based agent (KBA) uses knowledge stored in a knowledge base (KB) to make decisions

Component:

1) Knowledge Base (KB) :- Stores facts and rules
2) Inference Engine :- Applies reasoning to derive conclusions.
3) Perception (sensors) - collects environmental data.
4) Actuators :- Executes actions.
5) Learning Module - Updates KB overtime.

Ex: An expert medical diagnosis system that suggests treatment based on stored medical knowledge.

Learning Agent Architecture:

A learning agent improves its performance over time by learning from past experiences.

Components:

1) Learning Element - Learns patterns from past actions
2) Performance Element - Makes decisions based on learned data.
3) Critic - Evaluates agent performance and give feedback
4) Problem Generator - Suggest new experiences for learning.

Ex: A self driving car that refines its driving skills through real-world experience.

Convert the following to predicates:

a) Anita travels by car if available otherwise travels by bus

Available(Car) → Travels (Anita, Car)

¬ Available (Car) → Travels (Anita, Bus)

b) Bus route includes Andheri and Goregaon

Route (Bus, Andheri) ∧ Route (Bus, Goregoan)

c) Car has a puncture and is unavailable.

Puncture (Car) → ¬ Available (Car)

Will Anita travel via Goregaon.

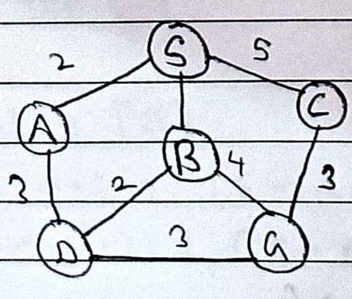
Forward Reasoning:

Given: Puncture (Car), we infer ¬Available (Car)

Thus, Travels (Anita, Bus)

Since, the bus route includes Goregaon, Anita will travel via Goregaon.

Q10.] Find the route from S to G using BFS.



→ Steps for BFS :

1) Start at node S, mark it visited.
Enque S

S

ii) Dequeue S and explore its neighbours (e.g. A, B, C)

	Queue (Q)
	Processed (P)

(i)

S			Q
			P

(ii)

A	B	C	Q
S			P

(iii)

B	C	D	Q
S	A		P

(iv)

C	D	G
S	A	B

(v)

D	G			Q
S	A	B	C	P

(vi)

G					Q
S	A	B	C	D	P

(vii)

						Q
S	A	B	C	D	G	P

Ajacency list $S \to \{A, B, C\}$
$A \to \{D\}$
$B \to \{D, G\}$
$C \to \{G\}$
$D \to \{G\}$

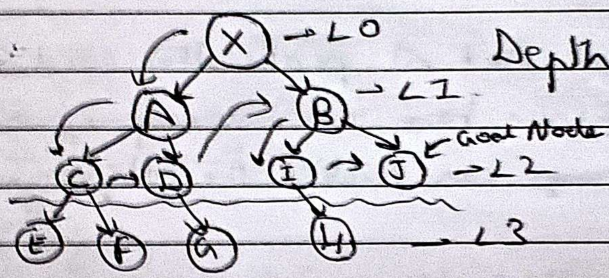
from BFS and adjacency list
shortest path is $S \to B \to G$
other paths are $S \to C \to G$ and $S \to B \to D \to G$ and $S \to A \to D \to G$.

Q 11) What do you mean by depth limited Search? Explain Iterative Deepening Search with example.

→ Depth-limited Search Algorithm:

- Working is similiar to DFS but with a predetermined limit.
- Helps in solving the problem of DFS: Infinite Path

Example:

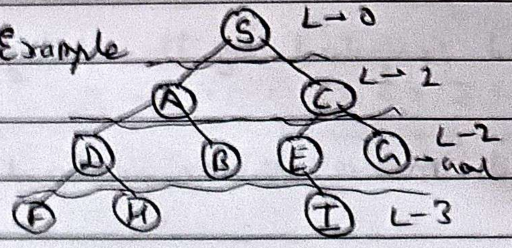


Depth d = 2

Path: X → A → C → D → B → I → J

- Iterative Deepening Depth First Search
  ↳ Combination of both DFS and BFS
  ↳ Best Depth Limit is found out by gradually increasing limit

Example



1st Iteration, d = 0 [S]

2nd Iteration, d = 0+1 = 1 [S → A → C]

3rd Iteration, d = 1+1 = 2 [S → A → D → B → C → E → G]

Q12.] Explain Hill climbing and its drawbacks in detail with example. Also state limitations of steepest ascent hill climbing

→ Hill Climbing: A local search algorithm that moves towards the highest value neighbouring state.
Ex: finding the highest peak in a mountain range.

Drawbacks:
• Local Maxima: May get stuck at peaks that aren't the best.
• Plateau: If all neighbours have same value progress halts
• Ridges: Slanted pathways can mislead the algorithm.

~~Steepest Ascent Hill Climbing: Chooses the best possible move at each step but fails if no better move ~~exists~~ exists.~~

Q13.] Explain Simulated Annealing and write its algorithm.

→ A probabilistic search method inspired by the cooling process of metals.
Allows occasional downhill moves to escape local optima.
Algorithm:
i) Start with an initial solution & temperature T.
ii) Generate a neighbouring solution.
iii) If its better, accept it; otherwise accept it with probability $e^{(-\Delta E/T)}$
iv) Decrease T gradually and repeat until T is very small.

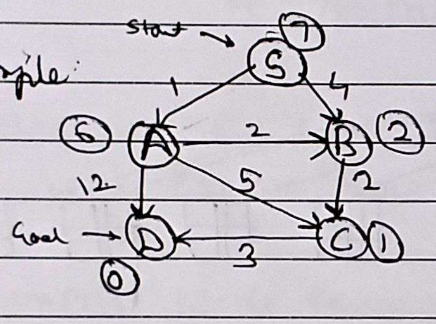
Use Case: Travelling Salesman Problem (Optimization problems)

Q14.] Explain A* Algorithm with an example

→ Uses heuristic funct^n h(n) and cost to reach the node 'n' from state.

$$\underbrace{f(n)}_{\text{estimated cost}} = \underbrace{g(n)}_{\text{cost to reach node}} + h(n) \longrightarrow \text{heuristic value}$$

Example:



S → A = 1 + 6 = 7 ; S → B = 4 + 2 = 6 ; S → B → C = 4 + 2 + 1 = 6

$\boxed{S \to B \to C \to D = 4 + 2 + 3 + 0 = 9}$

S → A → B = 1 + 2 + 2 = 5 ; S → A → C = 1 + 5 + 1 = 7

$\boxed{S \to A \to D = 1 + 12 = 13}$

S → A → B → C = 1 + 2 + 2 + 1 = 6

$\boxed{S \to A \to B \to C \to D = 1 + 2 + 2 + 3 = 8}$

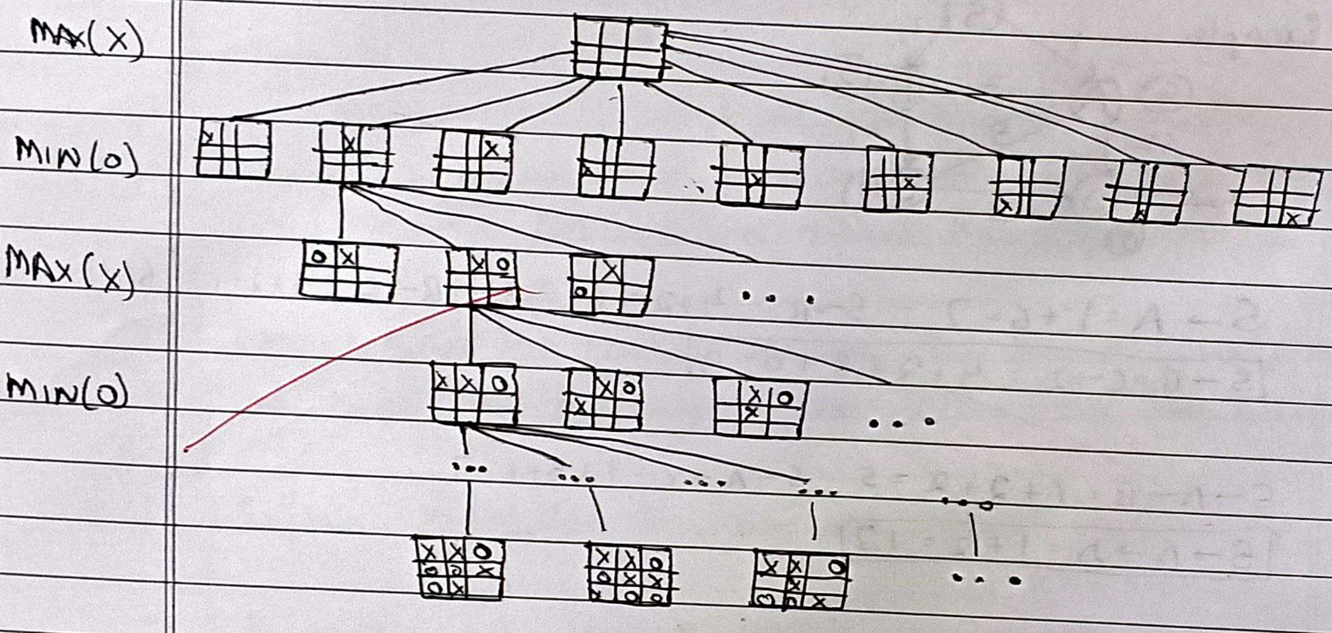
$\boxed{S \to A \to C \to D = 1 + 5 + 3 = 9}$

∴ The shortest path is $\boxed{S \to A \to B \to C \to D}$ with path cost 8.

Q15.] Explain Min Max algorithm and draw game tree for Tic Tac Toe

→ The Minimax algorithm is a decision making and game theory strategy used to find the best move for a player assuming the opponent also plays optimally.

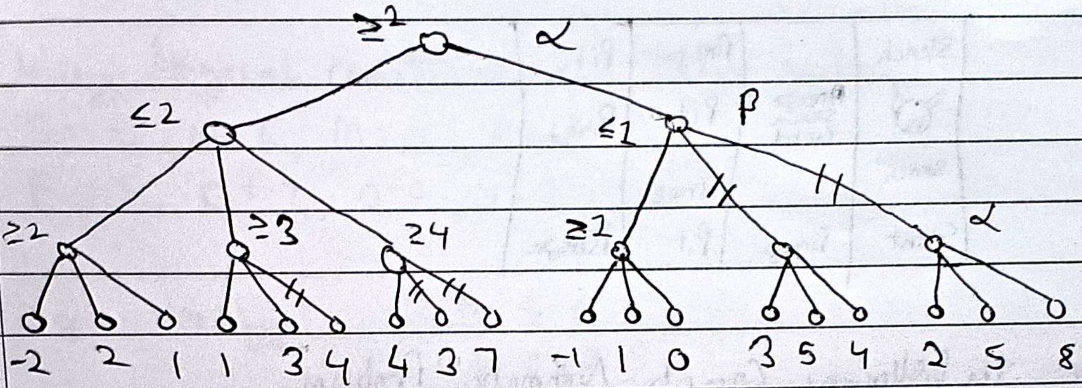
TicTacToe's Game Tree:



Q16.] Explain Alpha Beta pruning algorithm ~~and~~ for adversial search with example.

→ Alpha-Beta Pruning: Optimizes Minimax by eliminating unneccary branches

α (alpha): Best Value for MAX
β (beta): Best value for MIN
If $\boxed{\beta \le \alpha}$, further exploration is stopped (pruned).



$\alpha - \beta$ Pruning

Q17] Explain WUMPUS world environment giving its PEAS description. Explain how percept sequence is generated?

→ WUMPUS World Environment:

A grid based environment for AI agents.

PEAS.

- Performance: Gold collection, avoiding pits/wumpus
- Environment: Grid with wumpus, pits, gold.
- Actuators: Move, grab, shoot.
- Sensors: Breeze (pit), Stench (wumpus), Glitter (gold).

Percept Sequence:

- Move: To move in forward direction.
- Turn: To turn right by 90° degrees or left by 90° degrees.
- Grab: To pick up gold if it is in the same room as the player.
- Shoot: To shoot an arrow in a straight line in the direction faced by the player

Stench		Breeze	Pit
	Breeze Stench Gold	Pit	Breeze
Stench		Breeze	
Stent	Breeze	Pit	Breeze

Q18.] Solve the following Crypto-Arithmetic Problem
SEND + MORE = MONEY

→ Step ① : Assign unique digits to letters
- S, E, N, D, M, O, R, Y are distinct digits.
- S and M cannot be 0 (Since they are leading digits

Step ② :-

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

Step ③ :-

- Identifying M.

  Since M must be 1 (because S+M carries over).
- Finding O

  Since MORE contributes a carry to MONEY, the sum of SEND + MORE must be over 9999

  Only digit left for O that works is 0
- Determining S

  Since S+M = 10 and M=1, it means S = 9
- Finding other values

Using logical constraints and testing values,
$S = 9, N = 6, M = 1, R = 8$
$E = 5, D = 7, O = 0, Y = 2$

Step(4): Verify:

$$\begin{array}{r} 9\,5\,6\,7 \\ +\,1\,0\,8\,5 \\ \hline 1\,0\,6\,5\,2 \end{array}$$

The sum is 10652 which matches MONEY
$\therefore S = 9, E = 5, N = 6, D = 7, M = 1, O = 0, R = 8, Y = 2$

Q15] Consider the following axioms:
All people who are graduating are happy.
All happy people are smiling.
Someone is graduating.
Explain the following:
i) Represent the axioms in first order predicate logic.
ii) Convert each formula to clause form.
iii) Prove that "Is someone smiling?". Using resolution technique. Draw the resolution tree.

→ Step(1): Representing the axioms in first order predicate logic
$G(x)$: x is graduating
$H(x)$: x is happy
$S(x)$: x is smiling
Using these predicates, the axioms can be written as
(i) Axiom 1: "All people who are graduating are happy"
$\forall x (G(x) \rightarrow H(x))$

② Axiom 2 : "All happy people are smiling"
$\forall x (H(x) \rightarrow S(x))$.

③ Axiom 3 : "Someone is gradnating"
$\exists x\, G(x)$

Step ② :- Convert each formula to Clause Form:
Axiom 2 :- $\forall x\, (G(x) \rightarrow H(x))$
Convert implication
$\forall x\, (\sim G(x) \vee H(x))$
Convert to clause form
$\sim G(x) \vee (H(x))$ Clause 1
Axiom 2: $\forall x\, (H(x) \rightarrow S(x))$ becomes
$\sim H(x) \vee S(x)$ Clause 2
Axiom 3 : $\exists x\, G(x)$
Existential quantifier is eliminated.
$G(a)$, Clause 3

Step ③ :-
To prove that someone is smiling we need to show $\exists x S(x)$
using proof by contradiction, we assume the negation of the statement $\sim S(x)$

Applying resolution:
We start with known clauses
Clause 1 : $\sim G(x) \vee H(x)$
Clause 2 : $\sim H(x) \vee S(x)$
Clause 3 : $G(a)$

Negated Goal: $\sim S(a)$

Now we apply resolution step by step:

① Resolve (Clause 1) with (Clause 3):

$\sim G(a) \vee H(a)$

$G(a)$

Resolution: Remove $G(a)$ since it cancels $\sim G(a)$

New clause: $H(a)$

② Resolve (Clause 2) with $H(a)$:

$\sim H(a) \vee S(a)$

$H(a)$

Resolution: Remove $H(a)$ since it cancels $\sim H(a)$

New Clause: $S(a)$

③ Resolve $S(a)$ with $\sim S(a)$ (Negated Goal):

$S(a)$

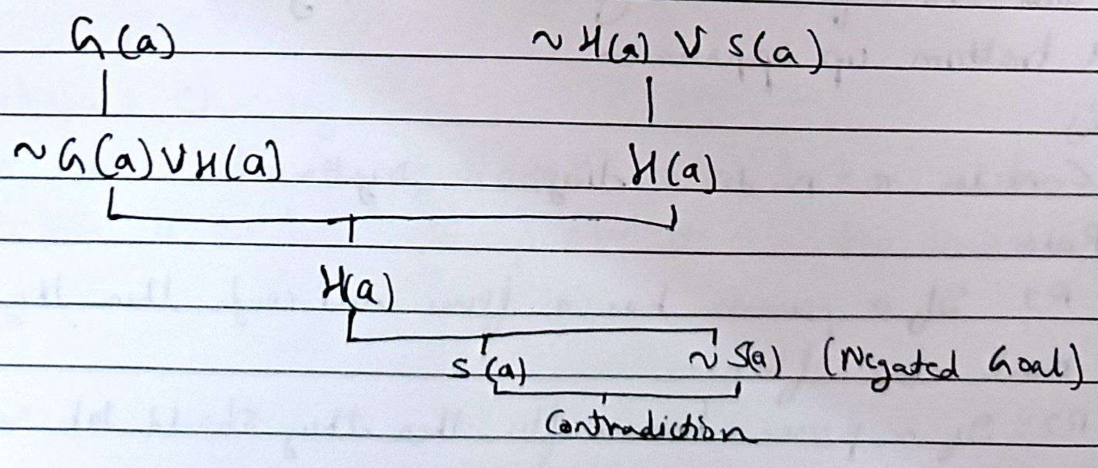
$\sim S(a)$

Contradiction (Empty clause)

Since we derived a contradiction, our initial assumption ($\sim S(a)$) is false, proving that $\exists x\ S(x)$

i.e Some one is smiling.

Step ④ :- Resolution Tree



Q 20.) Explain Modus Ponen with suitable example

→ Modus Ponens is a fundamental rule of inference in logic
It follows the structure:
① If P, then Q (Conditional Statement)
② If P is true (Premise)
③ Therefore, Q is true conclusion
Example:
① If it rains, the ground will be wet (If P, then Q)
② It is raining (P is true)
③. Therefore, the ground is wet (Q is true).

This rule is widely used in logical reasoning and mathematical proofs.

Q21.) Explain forward chaining and ~~logical~~ backward chaining algorithm with the help of example.

→ Forward Chaining:-
Forward Chaining starts with known facts and applies rules to infer new facts until a goal is reached. It is a bottom up approach.
Eg:
Consider a medical diagnosis system.
Rules:
R1: If a person has a fever and cough then they may have the flue.
R2: If a person has a flu, then they should take rest.

Facts:

1. A patient has a fever
2. The patient has a cough

Process:

1. The system checks R1: Since the patient has a fever and cough → They may have the flu.
2. The system check R2: Since the patient has the flu → They should take rest.

Conclusion :- The patient should take rest.

Algorithm:

1. Initialize: Start with a set of known facts in the KB
2. Match Rules: Identify rules whose conditions match known facts.
3. Apply rules: If a conditions are satisfied, infer the new fact (conclusion)
4. Update KB: Add the newly inferred fact to the KB.
5. Repeat: Continue the process until either:
   - The goal fact is derived or
   - No new facts can be inferred.

Backward Chaining:-

It starts with the goal and works backward to determine if there is evidence to support. It is a top down approach.

Algorithm:-

① Start with the goal:- Identify the target fact that needs to be proven.

② Check if the goal is already a known fact:-
- If yes, stop (goal is achieved)
- If no, proceed to next step

③ Find rules that conclude the goal:
- Identify rules where the goal is the conclusion.
- Check if the rules conditions are met.

④ Verify premises:-
- If all premises are known facts, apply rule and derive the goal.
- If some premices are unknown, set them as new subgoals and repeat the process.

⑤ Continue recursively until:
- The goal is proven (success), or
- No supporting facts are found (failure)

Example: diagnosing disease.

Goal: Determine if patient has flu.

Process: ① The system checks R1: "If a person has fever and cough, then they may have the flu".

② It asks "Does the patient have a fever?" → Yes

③ It asks "Does the patient have a cough?" → Yes

④ Since both conditions are met, the system confirms "The patient has flu"

* * *