Name:- Alok Yadav
Roll no: 59
Div: D15C

AIDS

Page No.
Date

05/05

Assignment - 1

1. 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) is a branch of computer science that enables machines to perform tasks that typically require human intelligence. These tasks include learning, reasoning, problem-solving, perception, language understanding and decision making.

(1) AI-powered algorithms analyzed news, social media, and official reports to detect early signs of outbreaks.

(2) Drug Discovery & Vaccine development: AI accelerated the development of COVID-19 treatments and vaccines by analyzing protein structures and predicting potential drug candidates.

(3) Medical diagnosis: AI powered diagnostic tools analyzed x-rays and CT scans to detect COVID-19 infections faster than traditional methods.

(4) Contact Tracing and spread prediction: AI models helped track virus spread using data from mobile apps, GPS, and social media.

(5) Remote Work & online learning: AI-powered collaboration tools (Zoom, Teams) enabled remote work and virtual meetings.

(6) Mental Health & Wealth being: AI chatbots like Wysa and Woebot provided mental health support to people dealing with stress and isolation.

2. What are AI agents terminology, explain with examp

2) An AI Agent is an entity that perceives its environment through sensors and acts upon it using actuators to achieve a specific goal. AI agents operate based on decision-making models, algorithms and data.

(1) Agent:
An agent is any system that perceives its environment and takes actions to achieve a goal.
example: A self-driving car perceives traffic signals and road conditions and takes actions like stopping and accelerating.

(2) Environment
The environment is everything that surrounds an AI agent and affect its performance.
example: In a chess-playing AI, the chessboard, opponent, moves, the rules form the environment.

(3) perception
perception is the agent's ability to gather information from the environment through sensors.
example: A facial recognition system perceives images through a camera.

(4) Sensors
Sensors are the input devices that collect data from the environment.
example:- A voice assistant (like Alexa) uses a microphone

(5) Actuators

Actuators are components that allow the agent to take action based on decisions

example: In a robot, it's arms & wheels are actuators

(6) Rational agent

A rational agent takes the best possible action to maximize its performance measure.

example: AI stock trading system makes decisions to maximize profit.

(7) Utility ~~agent~~ function

A utility function measures how good an action or state is for achieving a goal.

example: In a game AI, winning a match has a high utility score.

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

=) The 8-puzzle problem is a classic problem in artificial intelligence that involves sliding tiles on a 3x3 grid to reach a goal state.

Representation of the 8-puzzle problem

- Initial state: Any random arrangement of tiles
- Goal state: A predefined arrangement (eg. numbers in order)
- Operators: Move the blank tile (up, down, left, right)
- State space: All possible tile arrangements
- Cost function: Typically, each move has a cost of 1.

Several AI search techniques can be used to solve the 8-puzzle problem:

1) Uninformed search

a) Breadth-first search (BFS)

i) explores all possible moves level by level

ii) Guarantees finding the shortest solution if the solution exists

disadvantages: Requires high memory for large state spaces

b) Depth-first search (DFS)

explores one branch of the state space deeply before backtracking.

disadvantage: May get stuck in deep, infinite paths.

c) Iterative deepening Depth-first-search

Combines BFS and BFS to avoid deep recursion

2) Informed search Methods (Heuristic search)

a) ~~Greedy~~ Best-first search

i) Uses heuristic function to expand the most promising state first

ii) Heuristic example: Number of misplaced tiles.

b) A* search

i) combines path cost and heuristics $f(n) = g(n) + h(n)$

where $g(n)$: cost from the start state to current state

$h(n)$: Estimated cost from current state or goal.

Page No.
Date

common heuristics for A*

(1) Misplaced Tile Heuristic (h_1)

↳ Counts the number of misplaced tiles.

(2) Manhattan distance (h_2)

↳ Measures the sum of distances each ~~tile~~ tile is from its goal position.

↳ More accurate than misplaced tiles heuristic

4. What is PEAS descriptor? Give PEAS descriptor for following:

i) ~~Taxi~~

=) The PEAS (performance measure, Environment, Actuators, Sensors) descriptor is used to define the components of an Agent, helping to analyze its working environment and functions.

Performance Measure (P): The criteria for evaluating the agent's success

Environment (E): The external surroundings where the agent operates.

Actuators (A): The mechanisms through which the agent interacts with the environment

Sensors (S): The device used to perceive the environment

i) Taxi driver.

P: safety, fuel efficiency, passenger satisfaction

E: Roads, traffic signals, other vehicles, pedestrians, weather

A: Steering wheel, accelerator, brakes, horn, indicators, wipers

S: GPS, cameras, LiDAR, speed sensors, fuel gauge, odometer

ii) Medical diagnosis system

P: Diagnosis, accuracy, speed of response, patient recovery rate

E: patients, symptoms, medical databases, hospital environments

A: Display screen, speaker (for communication with doctor and patients)

S: patients history, test reports, X-rays, MRI scans, symptom inputs

iii) A Music composer.

P: quality of generated music, creativity, audience engagement

E: Musical database, sound libraries, listener preferences

A: Music output system (MIDI, speakers, digital files)

S: User feedback, musical trends, mood analysis, input instruments

iv) An aircraft autolander

P: smooth and safe landing, fuel efficiency, passenger comfort

E: Runway, weather conditions, altitude, air traffic

A: flaps, landing gear, engine throttle, air brakes

S: Gps, altimeter, wind sensors, gyroscopes, radar.

v) An essay evaluator

P: Accuracy in grading, fairness, grammer and coherence detection

E: Essays, answer sheets, writing rules, academic guidelines

A: Score display, feedback generator

S: Optical character recognition (OCR), NLP tools.

vi) Robotic sentry gun for the tech lab

P: Accuracy in target detection, response time,

E: lab premises, intruders, authorized personnel, obstacles

A: Gun turret, alarm system, movement motors

S: Motion detectors, thermal cameras, facial recognition, infrared sensors.

5. Categorize a shopping bot for an offline bookstore according to each of the six dimensions (fully/partially observable, deterministic/stochastic, episodic/sequential, static/dynamic, discrete/continuous, single/multi-agent)

=)

Observability: partially observable.

The bot may not have complete information about books on shelves, customer preferences or stock updates without external input.

Deterministic vs Stochastic :- Stochastic

Book availability may change due to manual sales, external purchases, or misplaced books, making environment uncertain

Episodic vs Sequential : Sequential

Each customer interaction affects the next steps (eg. book recommendations depend on previous queries).

Static vs Dynamic :- Dynamic

The environment changes (books sell out, new books arrive, customer preferences shift) making it dynamic

discrete vs continuous :- discrete
The bot deals with a finite set of actions (searching books, checking stock).

Single-Agent vs Multi Agent : Multi-Agent.
The bot interacts with multiple customers, bookstore staff, and possibly other inventory systems.

6. Differentiate Model based and Utility based Agent.

=>

Model-Based Agent	Utility-Based Agent.
(1) Uses an internal model of the environment to make decisions.	(1) Uses a utility function to measure the desirability of different states and select the best action.
(2) Chooses actions based on a representation of how the world works.	(2) Chooses actions that maximize its expected utility.
(3) Maintains a model of the environment (how actions affect future states)	(3) Uses a utility function to compare possible outcomes.
(4) focuses on achieving a goal using state transition models	(4) focuses on maximizing long-term benefits rather than just reaching a goal
(5) e.g. A self driving car that predicts traffic patterns & plan routes	(5) A stock trading bot that evaluates different portfolios to maximize returns

7. Explain the architecture of knowledge based agent and Learning Agent.

=>

Architecture of Knowledge-Based Agent

A knowledge-based agent (KBA) is an AI system that uses stored knowledge to make informed decisions. It consists of the following components

① Knowledge Base (KB)
It stores facts, rules and heuristics about the world

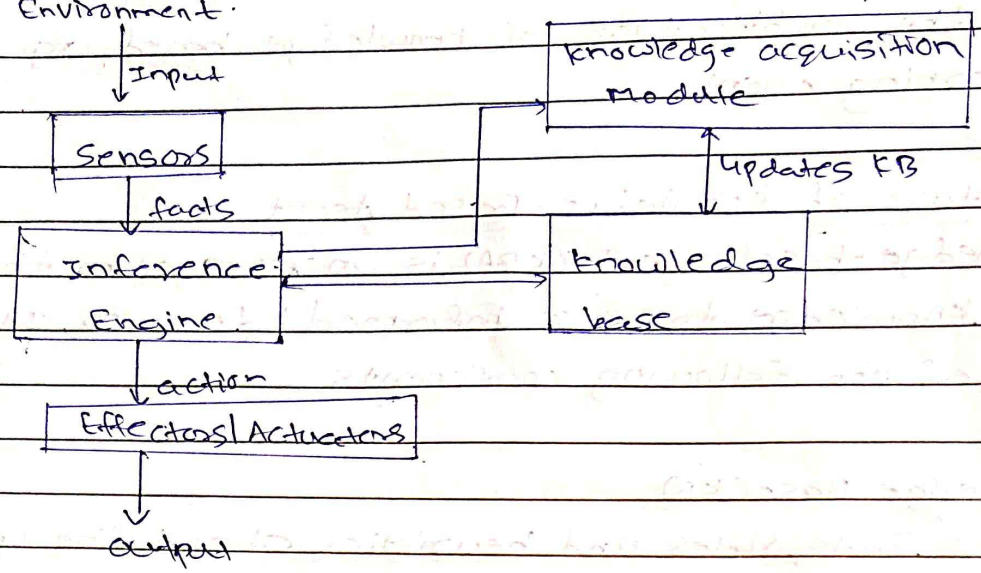
② Inference Engine
It Applies logical reasoning to derive new knowledge from stored facts.

③ perception (sensors)
It collects information from the environment

④ Actuators
performs actions based on inferences

⑤ Knowledge Acquition module
updates and expands the knowledge base with new data

Working process:
① The agent perceives the environment
② It queries the knowledge base for relevant information
③ The inference Engine applies logic to decide an action
④ The action is executed, and KB is updated if needed



Architecture of Learning Agent

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

Components of Learning Agent

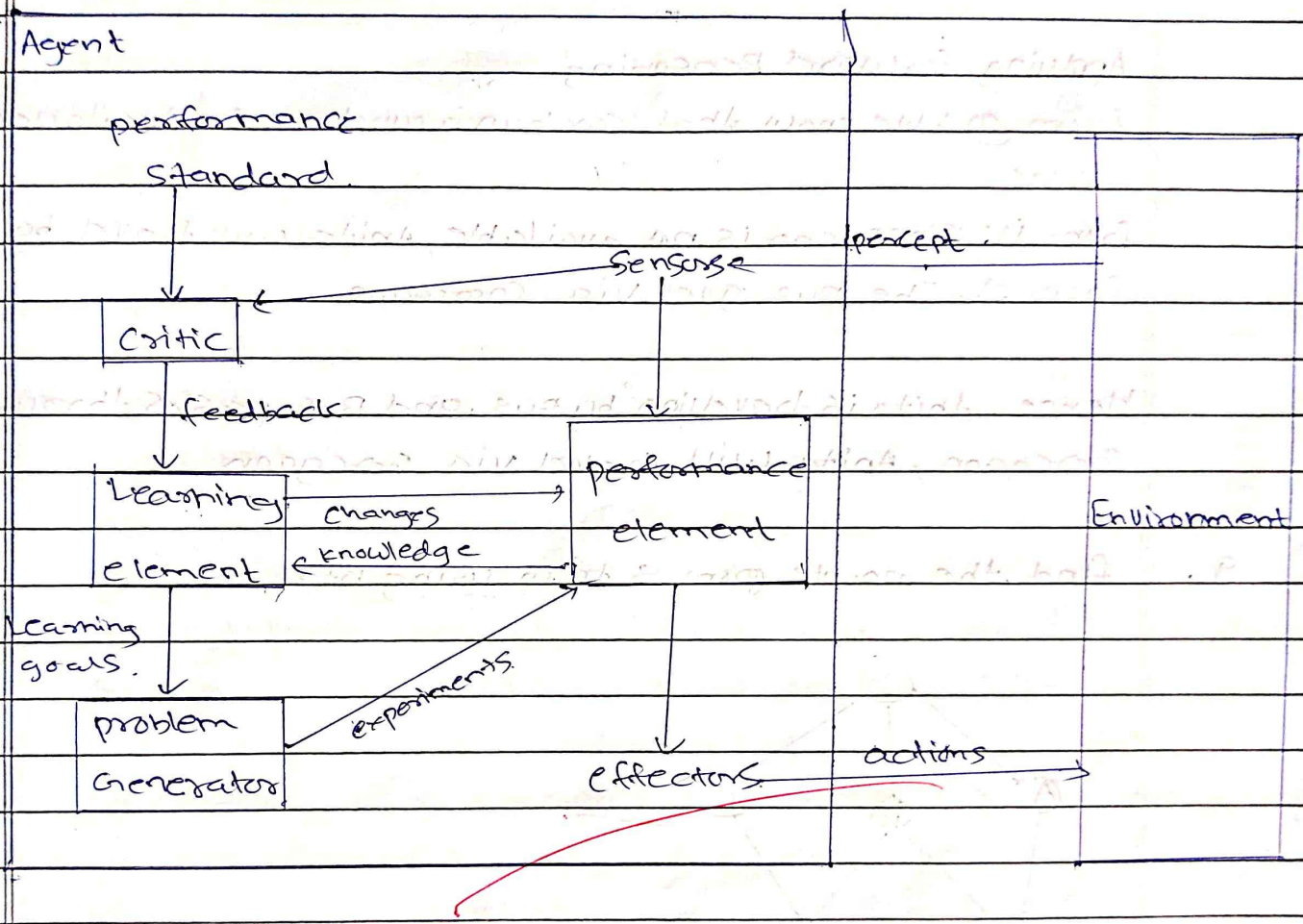
1. Learning Element: Improves the agent's knowledge by analyzing past experiences
2. performance Element: chooses actions based on the learned knowledge
3. critic: Evaluates the agent's actions by comparing outcomes with expected results
4. problem generator: suggests new exploratory actions to improve learning

Working process:

1. The performance element makes decisions and takes actions.
2. The critic evaluates the results and provides feedback

(3) The learning Element updates the knowledge based on feedback

(4) The problem Generator suggests new Strategies to improve performance.



8. Convert the following to predicates

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

Available(car) → Travels(Anita, car) — (1)

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

b) Bus goes via Andheri and Goregaon.

GoesVia(Bus, Andheri) ∧ GoesVia(Bus, Goregaon) — (2)

c) car has a puncture, so it is not available

puncture (car) → ¬Available (car) -③

Will Anita travel via Goregaon

Applying forward Reasoning.

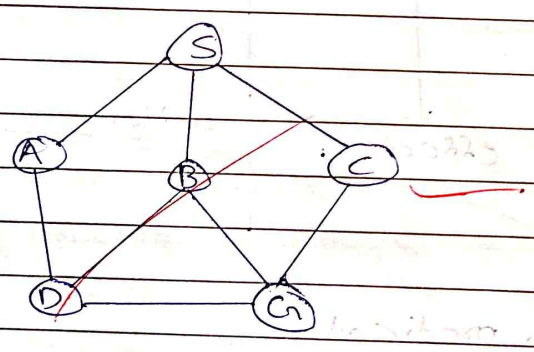
from ③ : We know that car has a puncture, so Available(car) is false.

from ① : since car is not available, Anita will travel by bus

from ② : The Bus goes via Goregaon.

Hence, Anita is travelling by Bus and Bus passes through Goregaon, Anita will travel via Goregaon.

9. find the route from S to G using BFS

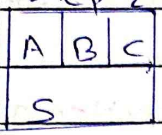


	Queue (Q)
	processed (P)

Step 1:

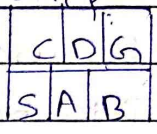
S

Step 2

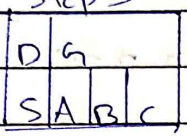


Step 3

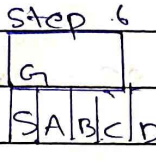
Step 4



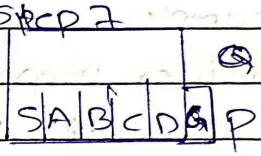
Step 5



Step 6



Step 7



Adjacency list

S → {A, B, C}
A → {D}
B → {B, G}
C → {G}
D → {G}

from BFS and adjacency list
shortest path is S → B → G
other paths are S → C → G and S → B → D → G and S → A → D → G

10. What do you mean by depth limited search? Explain Iterative Deepening search with example

⇒

Depth-Limited search (DLS)

Depth-limited search is a variation of depth-first search (DFS) where we impose a depth limit to avoid going too deep into an infinite or large search space.

How DLS works

① The algorithm follows a depth-first search (DFS) strategy but limits the depth of recursion

② If the goal is found within the limit, the search stops

Page No.
Date

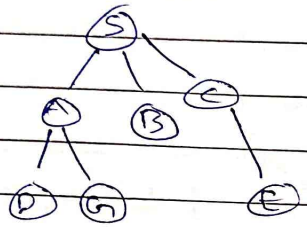
example

③ If the goal is not found within the limit, it returns failure or cutoff

④ This helps in avoiding infinite loops in problems with large or infinit depth.

example:

consider a graph where we want to find a path from S to G with depth limit of 2.



↳ If the depth limit is 1, we only explore S→A, B, C but cannot reach G

↳ If the depth limit of 2, we explore S→A→D, G.

Iterative deepening search(IDS)

Iterative deepening search (IDS) combines the benefits of depth-first search (DFS) and Breadth-first search (BFS) It repeatedly performs DLS with increasing depth limits until the goal is found.

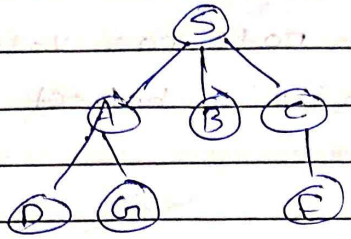
How IDS works:

① start with depth limit = 0 and perform DLS

② Increase the depth limit & perform DLS again

③ Repeat until goal is found.

example:



a) Depth limit 0 → only node S is checked
b) Depth limit 1 → explores A, B, C, but G is not found
c) Depth limit 2 → explores D, G, E and finds G

11. Explain Hill climbing and its drawback in detail with example. Also state limitations of steepest-ascent hill climbing

:-)

Hill climbing is an optimization algorithm that continuously moves towards higher-valued states (i.e. better solutions) until a peak (local maximum) is reached. It is a greedy algorithm that evaluates neighbouring states and chooses the one with the highest value. It is widely used in Artificial Intelligence, especially in problems like pathfinding, scheduling problems

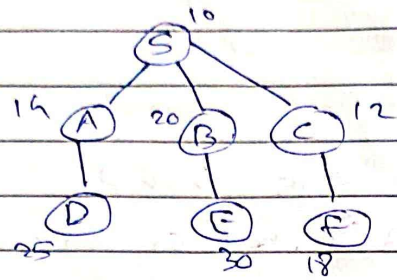
working of Hill climbing.

1. Start with an initial state (solution).
2. Evaluate the neighbouring states of the current state
3. Move to the best neighbouring state that improves the solution
4. Repeat the process until no better neighbour is found or predefined goal state is reached.

Page No.
Date

example:

consider a graph where each node has value representing its height and goal is to find the highest valued node



(1) Start at node S (10)

(2) check its neighbors: A, B, C then move to B (highest value = 20)

(3) check neighbors of B: F(30) then move to F (highest value = 30)

(4) F has no better neighbour, so stop

(5) final result: Node F (value = 30) is the peak

Drawbacks of Hill climbing:

(1) Local Maxima: if node D(25) was chosen from A instead of B, it would be stuck there, not reaching 30.

(2) plateaus: If multiple nodes had the same value, the algorithm might get confused

(3) Ridges: The algorithm cannot take downward moves to explore better paths.

(4) No backtracking: Hill climbing does not remember previous states, so if it gets stuck, it cannot backtrack to explore better paths

Page No.
Date

Steepest-Ascend Hill climbing and its limitations

Steepest-Ascent hill climbing is a variation where the algorithm evaluates all neighbouring states and moves to the one with the highest improvement.

Limitations

(1) Since it evaluates all neighbours, it takes more time and resources

(2) even though it selects the best move at each step. it cannot escape local maxima

(3) fails in plateaus and Ridges

12. Explain simulated annealing and write its algorithm.

=>

Simulated Annealing (SA) is an optimization algorithm inspired by metallurgical annealing, where materials are heated and then cooled to remove defects. It helps escape local maxima by allowing occasional downward moves to explore better solutions

Working:

(1) Start with an initial solution

(2) Set a high temperature (T), which gradually cools down

(3) Select a random neighbor of the current solution

(4) Calculate the energy difference (ΔE) between the new and current solution.

i) If the new solution is better, accept it.

ii) If the new solution is worse, accept it with a probability: $p = e^{-\Delta E/T}$ where e is Euler's number,

T is the current temperature

(5) Reduce the temperature gradually.

(6) Repeat until the temperature is very low or a stopping condition is met.

Algorithm:

Input : Initial state S, Initial temperature T, cooling rate α, Stopping temperature T_min

Output : Best found Solution S_best

1. Set current_state ← S
2. Set current_cost ← cost_function(current_state)
3. Set best_state ← current_state
4. Set best_cost ← current_cost.
5. While T > T_min do:
   6. Generate a neighbor new_state of current_state
   7. Compute new_cost ← cost_function(new_state)
   8. Compute ΔE ← new_cost - current_cost
   9. If ΔE > 0 then
      10. Accept new_state as the current state
      11. Update current_state ← new_state
      12. Update current_cost ← new_cost
   13. Else
      14. Accept new_state with probability p = exp(ΔE/T)
      15. If random(0,1) < p then
         16. Update current_state ← new_state
         17. Update current_cost ← new_cost
   18. If current_cost > best_cost then
      19. Update best_state ← current_state

Page No.
Date

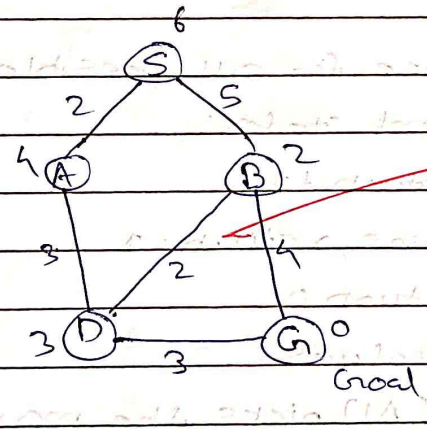
20. Update best_cost ← current_cost
21. Reduce temperature : T ← α * T
22. Return best_state

B. Explain A* Algorithm with an example

=>

A* is widely used graph search and pathfinding algorithm that finds the shortest path between a start node and a goal node. It is an informed search algorithm that uses both
↳ cost to reach the node (g(n))
↳ Estimated cost from the node to the goal h(n)
formula : f(n) = g(n) + h(n)

example :



Steps
↳ Initialize open list
↳ expand node with lowest f(n)
↳ update g(n), h(n) & f(n) for neighbouring
↳ Repeat until goal node is reached

Steps

Node	g(n)	h(n)	f(n) = g(n) + h(n)
start(S)	0	6	6
Expand A (S→A, cost = 2)	2	4	6
Expand B (S→B, cost = 5)	5	2	7
Expand D (A→D, cost = 2+3 = 5)	5	3	8
Expand G (B→G, cost = 5+4 = 9) (Goal reached)	9	0	9

14. Explain MiniMax Algorithm and draw game tree for Tic TACTOE Game

=>

Minimax is a decision-making algorithm used in two-player games like Tic-Tac-Toe, chess and connect four. It helps in finding the best possible move for a player by assuming that the opponent also plays optimally. The algorithm alternates between Maximizing (for AI) and minimizing (for opponent) and each state has a value

+1 → AI wins

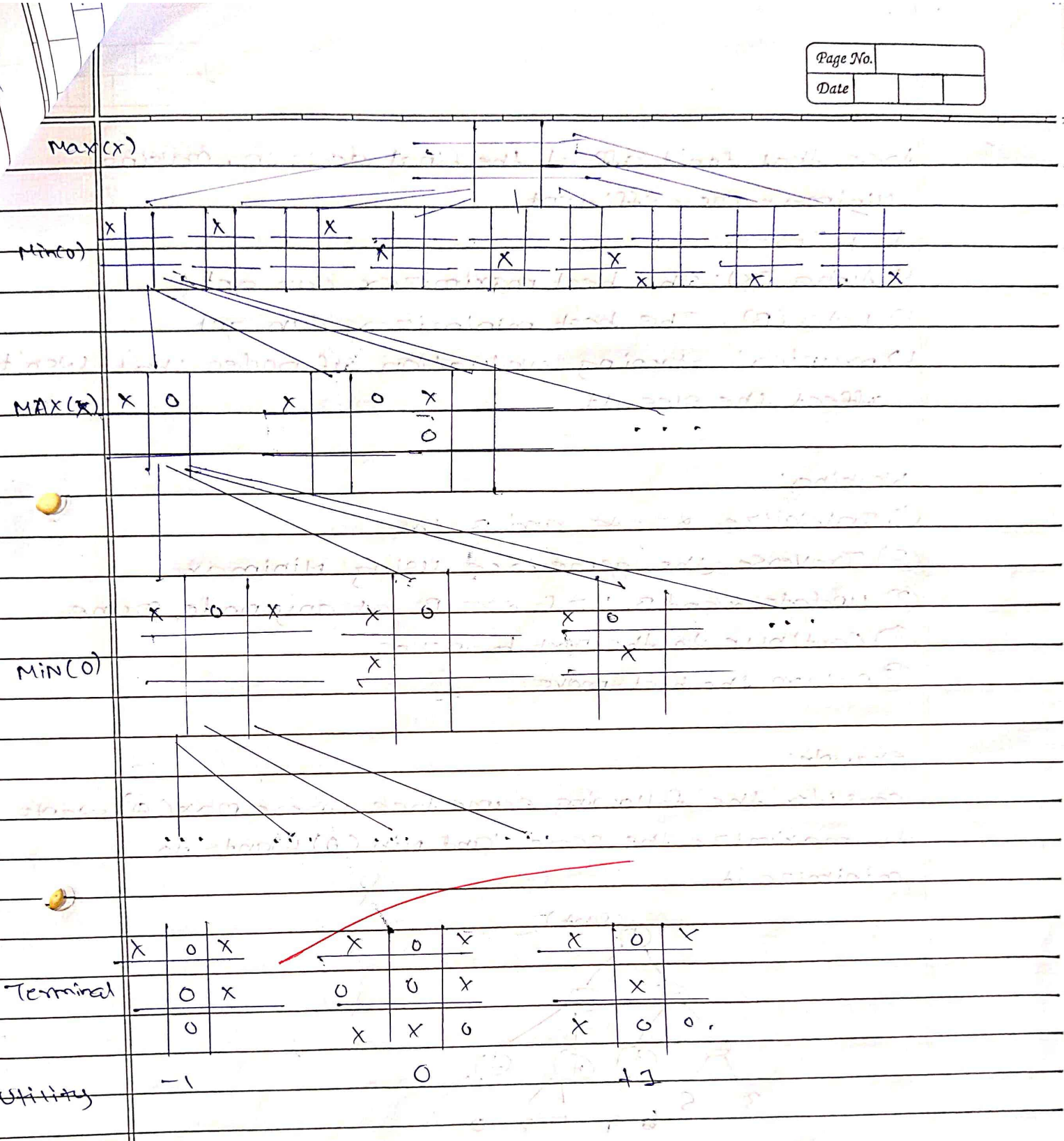
-1 → opponent wins

0 → Draw.

Algorithm steps:

1. Generate game tree for all possible moves
2. Evaluate Terminal states:
   - If AI wins, return +1
   - If opponent wins, return -1
   - If draw, return 0
3. Backpropagate values:
   - Max player (AI) picks the maximum value
   - Min. player (opponent) picks the minimum value
4. Select the best move at the root level

Game Decision Tree :-



15. Explain Alpha-beta pruning algorithms for adversarial search with example.

⇒ Alpha-beta pruning is an optimization technique for the Minimax algorithm. It eliminates branches in the game

tree that don't affect the final decision, making Minimax more efficient.

Key Terms:

↳ Alpha (α): The best maximizer can get

↳ Beta (β): The best minimizer can get

↳ pruning: stopping evaluation of nodes that won't affect the result.

Working:

① Initialize $\alpha = -\infty$ and $\beta = +\infty$

② Traverse the game tree using Minimax

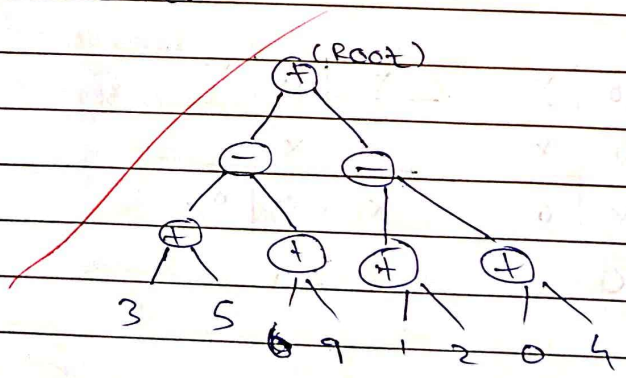
③ update α and β: If $\alpha >= \beta$ at any node, prune

④ Continue to the next branches

⑤ Return the best move

example:

Consider the following game tree where max (⊕) wants to maximize the score, and Min (⊖) wants to minimize it.



① Initialize: Alpha (α) $= -\infty$, Beta (β) $= +\infty$

② Evaluate left subtree

↳ Min (⊖) chooses between (3, 5) → selects 3 (smallest value)

↳ Max (⊕) chooses between (3, 6) → selects 6 (largest value)

↳ update x = 6

(3) Evaluate Right subtree:

↳ first node evaluated is 7

↳ Beta is updated β = 7

↳ Since x(6) ≥ β(7), we prune (skip checking 2, 0 and 4)

16. Explain WUMPUS world environment giving it PEAS description. Explain how percept sequence is generated?

=>

The Wumpus world is a grid-based environment used in Artificial Intelligence (AI) to demonstrate logical agent-based reasoning. It is a partially observable, stochastic, sequential environment where an AI agent must navigate a cave-like world while avoiding dangers.

Structure of Wumpus World

(1) Grid-Based (4×4 or larger)

(2) Contains:

↳ Gold (Goal: pick it up)

↳ Wumpus (A monster that kills the agent)

↳ Pits (If agent falls, it dies)

↳ Walls (Boundaries of the world)

(3) Sensory perceptions (percepts)

↳ Breeze → Near a pit

↳ Stench → Near Wumpus

↳ Glitter → Gold is nearby

↳ Bump → Hits a Wall

↳ Scream → Wumpus is dead.

PEAS description.

P : +1000 for finding gold, -1000 for falling into a pit or encountering wumpus, -1 for every move, -10 for shooting the arrow

E : A 4x4 grid with the agent, wumpus, pits, gold and walls

A : Move (Up, Down, Left, Right), Grab (Gold), Shoot (Arrow), Climb (Exit)

S : Breeze, Stench, Glitter, Bump, Scream.

How percept sequence is generated?

A percept sequence is a history of all sensor inputs received by the agent over time.

example:

(1) Agent starts at (1,1) → percepts = [Breeze] (pit is nearby)

(2) Moves to (1,2) → percepts = [Breeze, Stench] (pit + near wumpus)

(3) Moves to (2,2) → percepts = [Stench] (near wumpus)

(4) Moves to (3,2) → percepts = [Glitter] (gold is nearby)

Agent collects percepts at each step and makes logical decisions based on past percepts to avoid dangers and find gold.

17. Solve the following crypto-arithmetic problems.

SEND + MORE = MONEY

=>

5 4 3 2 1

S E N D

M O R E

C_3 C_2 C_1

M O N E Y

(1) From column 5, M=1, since it is only carry-over possible from sum of 2 single digit number in column 4

(2) To produce a carry from column 4 to column 5 'S+M' is atleast 9 so 'S = 8 or 9' so 'S+M = 9 or 10' and 'O = 0 or 1' But 'M=1', so 'O=0'.

(3) If there is carry from column 3 to 4 then E=9 and N=0. But O=0 so there is no carry and S=9 and C_3=0

(4) If there is no carry from column 2 to 3 then 'E = N' which is impossible, therefore there is carry and N = E+1 and C_2 = 1.

(5) If there is carry from column 1 to 2 then N+R = E mod 10 and N = E+1 so E+1R = E mod 10 so R=9 but S=9 so there must be carry from column 1 to 2. Therefore C1 = 1 and R=8.

(6) To produce carry C1 = 1 from column 1 to 2, we must have D+E = 10+Y as Y cannot be 0/1 so D+E is atleast 12. As D is atmost 7 and E is atleast 5 (D cannot be 8 or 9 as it is already assigned). N is atmost 7 and N = E+1 so E = 5 or 6.

(7) If E were 6 and D+E atleast 12 then D would be 7, but N = E+1 and N would also be 7 which is impossible Therefore, E=5 and N=6

(8) D+E is atleast 12 for that we get D=7 and Y=2

Hence final values:

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

```
 9567          SEND
+1085    =>  + MORE
-----        -----
10652         MONEY
```

18. Consider the following axioms.

i) All people who are graduating are happy.

ii) All happy people are smiling.

iii) Someone is graduating.

Explain the following

1) Represent these axioms in first order predicate logic

i) $\forall x$ (Graduating(x) → Happy(x))

ii) $\forall x$ (Happy(x) → Smiling(x))

iii) $\exists x$ Graduating(x)

2) convert each formula to clause form using identity $p \rightarrow q = \sim p \vee q$

i) ~Graduating(x) ∨ Happy(x) — (1)

ii) ~Happ(x) ∨ Smiling(x) — (2)

iii) Graduating(A) — (3)

3) prove that "Is someone smiling?" using resolution technique. Draw resolution tree.

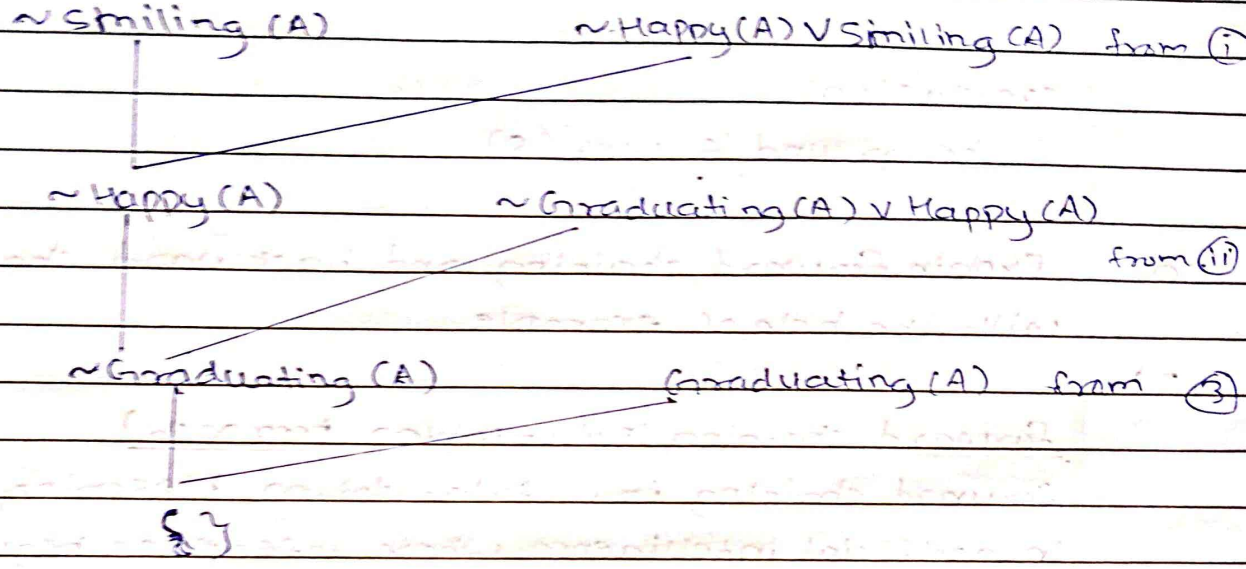
from (2), substitute x = A (since Graduating(A) is given).

∴ ~Happy(A) ∨ Smiling(A) — (i)

Page No.
Date

From (i), substitute x = A ∴ ~Graduating(A) V Happy(A) -(ii)

Resolution tree



As we get null subset our assumption of not Someone smiling is wrong. Hence Someone is smiling.
Hence "Is Someone Smiling" becomes True.

Q. Explain Modus ponen with suitable example.

=>

Modus ponen is a fundamental rule of inference in logic which states that: If we have:

(1) A conditional statement: p->Q (If p, then Q)

(2) p is true

Then we conclude that Q is also true

Symbolic Representation

(1) p->Q

(2) P (p is true) ∴ Q (Therefore, Q is true)

example:

premises

① If it rains, the ground will be wet ($p \rightarrow Q$)

② It is raining (p)

conclusion:

∴ The ground is wet (Q)

20. Explain forward chaining and backward chaining with the help of example.

=>

forward chaining (Data-Driven Approach)

forward chaining is a data-driven reasoning approach in artificial intelligence where inference begins with known facts and applies logical rules to derive new conclusions until a desired goal is reached. This method is often used in expert systems, rule-based systems and automated reasoning applications.

Algorithm:

① Start with known facts in the knowledge base

② Find rules where premises (conditions) match known facts

③ Apply the rule, infer new facts

④ Repeat until the goal is reached or no more rules apply.

example: Medical diagnosis

let's assume we have these rules

① If a person has a cough & fever, THEN they might have flu

Page No.
Date

② If a person has flu, THEN they should take rest and fluids.

Given facts
The patient has cough & fever.

forward chaining process:
① From rule ① , we infer flu
② From rule ②, we infer the patient should take rest and fluids.

Conclusion: The system recommends rest and fluids

Backward chaining (Goal-driven Approach)
Backward chaining is a goal-driven reasoning that starts with the objective and works backward to determine if the given facts support it. Instead of processing all available data, it selectively applies rules to establish whether the goal can be derived from existing knowledge. This method is commonly used in theorem proving, AI-Based problem-solving and prolog-based expert systems.

Algorithm:
① Start with the goal (conclusion to be proven)
② Check if the goal is already a fact.
③ If not, find the rules where goal is in the conclusion.
④ Check if the premises (conditions) of the rule are true.
⑤ Repeat until the goal is proven or disproven.

Date

Example : Diagnosing flu

Goal : Does the patient have the flu ?

① To prove "flu", we check if (cough and fever) are true
② We ask if the patient has cough → yes
③ We ask if the patient has fever → yes
④ Since both conditions are met, flu is confirmed

Conclusion : The patient has the flu.