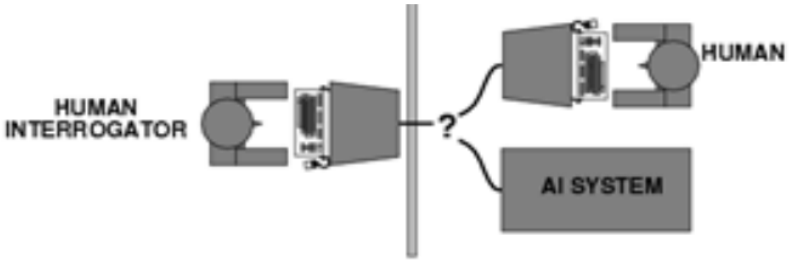
Turing Test:

- *What It Is*: The Turing Test, created by Alan Turing in 1950, is designed to determine if a machine can exhibit human-like intelligence.
- *How It Works:* In this test, a person has conversations with both a machine and a human without knowing which is which. If the person cannot tell them apart, the machine is considered to have passed the test and demonstrated intelligence similar to a human's.
- *Turing's Prediction:* Turing believed that by the year 2000, machines would be able to ***fool 30% of human judges*** in a five-minute conversation.



The Turing Test is a significant milestone in artificial intelligence (AI) as it sets a benchmark for machines' ability to interact like humans.

In ***AI, PEAS*** stands for *Performance, Environment, Actuators*, and *Sensors***.** This model helps us understand what an intelligent agent is supposed to do, the conditions it works in, and how it interacts with its surroundings. Here's a more detailed look at each element:

1. ***Performance****:*
    - This is a way to measure how well the agent is doing its job. We define specific goals or criteria to determine if the agent is successful.

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

- For example, in a vacuum cleaning robot, performance can be measured by how much dirt it collects, how well it avoids obstacles, or how efficiently it cleans a room without using too much power.
- ***Performance*** measures can vary depending on the task. For a self-driving car, it might be how safely it drives or how well it follows traffic laws.

2. ***Environment****:*
    - The **environment** is the world or setting in which the agent operates. It includes everything around the agent that it can interact with or be affected by.
    - The environment can be simple or complex. For a chess-playing AI, the environment is limited to the chessboard, pieces, and rules. However, for a self-driving car, the environment is much more complex, including roads, traffic signals, pedestrians, other vehicles, and weather conditions.
    - Understanding the **environment** helps the agent make informed decisions and adapt to changes.
3. ***Actuators****:*
    - **Actuators** are the physical components that allow the agent to take action. They enable it to interact with or affect its **environment**.
    - For example, a robot might have wheels to move, arms to pick up objects, or a vacuum motor to clean surfaces.
    - In a self-driving car, **actuators** could include the steering system, brakes, accelerator, and other controls that help the car navigate.
4. ***Sensors****:*
    - **Sensors** help the agent observe and gather information about the **environment**. They are like the agent's eyes, ears, and other senses.
    - For instance, a self-driving car uses cameras, radar, and LiDAR to detect obstacles, traffic signals, and road markings.

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

- In a smart thermostat, **sensors** detect room temperature and humidity. These inputs help the agent make better decisions and respond to changes in real-time.

***Summary**:*
The **PEAS** model helps break down the key parts of an AI agent's design. It answers questions like: *What is the agent's purpose? Where will it operate? How will it act? How will it perceive?* This model is useful for designing and evaluating AI systems in various applications, from robots and cars to games and home assistants.

In **AI**, an **Agent** is an entity that can perceive its environment, make decisions, and take actions to achieve specific goals. Agents use sensors to observe their environment and actuators to perform actions. They aim to make decisions that maximize their success in completing tasks.

Types of Agents

1. ***Simple Reflex Agents**:*
   - These **agents** respond directly to what they perceive, without considering past experiences.
   - They work well in simple environments where quick responses are needed.
   - Example: A basic thermostat that turns the heater on or off based on the current temperature.
2. ***Model-Based Reflex Agent*s**:
   - These **agents** keep track of some information about the world to handle partially observable **environments**.
   - They use a **model** (internal memory) to understand how the world changes.
   - Example: A cleaning robot that remembers where it has already cleaned.

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

3. ***Goal-Based Agents**:*
    - These **agents** act to achieve specific **goals**, not just to respond to immediate inputs.
    - They make decisions by comparing actions that bring them closer to their **goal**.
    - Example: A GPS system that finds the best route to reach a destination.

4. ***Utility-Based Agents**:*
    - These **agents** aim not just to achieve a **goal** but to maximize their happiness or "utility."
    - They use a utility function to compare different actions and choose the one with the highest satisfaction.
    - Example: A self-driving car that balances safety, comfort, and speed.
5. ***Learning Agents**:*
    - These **agents** improve over time by learning from their experiences.
    - They have components that let them adapt and perform better after every action.
    - Example: A recommendation system that learns user preferences and suggests better options over time.

Key Concepts Related to Agents

- **Perception**: Agents use sensors to perceive their environment. For example, a camera on a robot is a sensor.
- **Action**: Agents use actuators to perform actions. For instance, a robot's wheels act as actuators for movement.

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

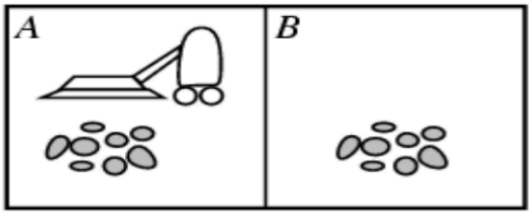
- **Environment**: The setting where the agent operates. It can be simple (like a board game) or complex (like real-world traffic).
- **Rationality**: A rational agent always tries to make the best decision based on its knowledge and goals.

Summary:

Agents are central to ***AI*** because they interact with their environment and make decisions. Each type of agent **i**s suitable for different tasks, from simple reactions to complex, goal-oriented learning. Understanding these types helps in designing AI systems that work effectively in various situations.

Vacuum-cleaner world

- **Percepts:**
  Location and status,
  e.g., [A,Dirty]
- **Actions:**
  Left, Right, Suck, NoOp



Example vacuum agent program:

function Vacuum-Agent([location,status]) returns an action

- *if* status = Dirty *then* return Suck
- *else if* location = A *then* return Right
- *else if* location = B *then* return Left

Machine Learning

Machine learning is a field of study focused on teaching computers to perform tasks by using data. Instead of programming the computer with specific instructions for every task, **machine learning** allows the computer to analyze data, identify patterns, and learn from experiences.

For example, if you want a computer to recognize images of cats and dogs, you would provide it with many labeled pictures of both animals. The

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

machine learning algorithm would then learn the differences between the two by examining the features of the images. Over time, it gets better at identifying new pictures without needing explicit instructions for each one.

Machine learning can be applied to many areas, such as:

- **Recommendation Systems**: Like those used by Netflix or Amazon, which suggest movies or products based on your past behavior.
- **Spam Detection**: Email services use **machine learning** to filter out spam by learning from previous spam messages.
- **Voice Recognition**: Virtual assistants like Siri or Google Assistant learn to understand spoken language through exposure to many different voices and accents.

In summary, **machine learning** enables computers to improve their performance on tasks through experience, making them more adaptable and efficient in handling complex problems.

Machine Learning

This is the study of how to make computers work by giving them data. This way, they can learn to do things on their own without being told exactly how to do it.

Neural Networks

These are groups of **algorithms** that are designed to work like the **human brain**. Neural networks help solve complicated machine learning problems.

Robotics

Robotics is a part of **AI** that focuses on robots. These robots are **artificial beings** that operate in the **real world**. An AI robot can **interact** with its surroundings by sensing, moving, and taking appropriate actions.

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

Expert Systems

An expert system is a computer program that imitates how a human makes decisions. It uses AI to simulate the judgment and behavior of someone with expert knowledge in a specific area.

Fuzzy Logic Systems

Fuzzy logic is a way of computing that deals with "degrees of truth" instead of just "true or false" (1 or 0) like regular computers. Fuzzy logic systems can understand unclear or noisy information.

Natural Language Processing (NLP)

NLP is an AI method that studies human language to gain useful insights and solve problems.

Artificial Intelligence *(AI)* is a way for machines to act like humans. Machine Learning (ML) is a part of AI. It involves teaching computers to do things by giving them data so they can learn on their own without needing specific instructions. In other words, machine learning is a method used to create artificial intelligence.

Here's a simplified explanation of how AI can help a bank manager determine loan approvals using the K-Nearest Neighbors (KNN) algorithm:

Using AI for Loan Approval Decisions

A bank manager has a large dataset with records of thousands of loan applicants. AI can assist the manager in deciding which loans to approve by using the KNN algorithm, which classifies loan requests into two categories: Approved and Disapproved.

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

1. ***Data Collection:***
    - Collect data about loan applicants, such as account balance, credit amount, age, occupation, and loan history.
2. ***Data Cleaning:***
    - Remove unnecessary information that doesn't help in predicting loan approvals, like telephone numbers.
3. ***Data Exploration & Analysis:***
    - Analyze data to understand relationships between different variables.
4. ***Building a Machine Learning Model:***
    - Use K-Nearest Neighbors (KNN) to classify loan applications.
5. ***Model Evaluation:***
    - Test how well the machine learning model works. Adjust settings if needed.

Conclusion:

By following these steps, AI can help the bank manager make informed decisions about loan approvals.

Here’s a simplified explanation of the differences between parametric and non-parametric models:

Parametric Models

1. ***Definition:*** Have a fixed number of parameters.
2. ***Examples:*** Common examples include:
    - Linear Regression: Models the relationship between variables using a straight line.
    - Logistic Regression: Used for binary classification problems, estimating probabilities.

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

3. ***Characteristics:***
    - The number of parameters is set before analyzing the data.

Non-Parametric Models

1. ***Definition:*** Do not have a fixed number of parameters.
2. ***Examples:*** Common examples include:
    - Decision Trees: Models that split data into branches based on feature values to make decisions.
    - K-Nearest Neighbors (KNN): Classifies data points based on the majority class of their nearest neighbors.
3. ***Characteristics:***
    - The number of parameters can grow with the dataset size.

Summary of Differences

- Parameters: Parametric models have a fixed number; non-parametric models do not.
- Flexibility: Parametric models are less flexible; non-parametric models are more adaptable.

Q-Learning

- Definition: A reinforcement learning algorithm that helps an agent learn the best actions.

Reinforcement Learning

- Definition: A type of machine learning where an agent learns to make decisions by receiving rewards or penalties.

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

TensorFlow

- Definition: An open-source library for building and training machine learning models.

Artificial Intelligence (AI)

- Definition: AI is about making computer systems that can think, learn, and solve problems like people do..

The four main sources that demonstrate human intelligence:

1. ***Problem-Solving Ability:*** The capacity to analyze, understand, and solve problems.
2. ***Learning and Adaptation:*** The ability to acquire knowledge and adjust behavior.
3. ***Language and Communication:*** Proficiency in expressing ideas.
4. ***Emotional Intelligence:*** Awareness and management of emotions.

Agent

Definition: An agent can sense its environment and take actions.

Rational Agent

Definition: An agent that makes the best decisions based on goals and knowledge.

Concise definitions:

1. **State**: A specific **situation** in a problem.
2. **State Space**: All possible **states**.
3. **Search Space**: Area where a **search algorithm** looks.
4. **Search Node**: A point in the **search space**.
5. **Goal**: The desired **outcome**.
6. **Action**: A **move** to change the current state.

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

7. **Transition Model**: Describes how **actions change states**.
8. **Branching Factor**: Average number of **actions** available from a state.

4. List the criteria to measure the performance of different search strategies. [3]

All search strategies

Algorithm	Complete?	Optimal?	Time complexity	Space complexity
BFS	Yes	If all step costs are equal	$O(b^d)$	$O(b^d)$
UCS	Yes	Yes	Number of nodes with $g(n) \leq C^*$	
DFS	No	No	$O(b^m)$	$O(bm)$
IDS	Yes	If all step costs are equal	$O(b^d)$	$O(bd)$
Greedy	No	No	Worst case: $O(b^m)$ Best case: $O(bd)$	
A*	Yes	Yes	Number of nodes with $g(n)+h(n) \leq C^*$	

b: maximum branching factor of the search tree
d: depth of the optimal solution
m: maximum length of any path in the state space
C*: cost of optimal solution

Searching Problem Components

1. **Start State**: This is the initial point where the agent begins its search process. It represents the starting condition or situation for the search.

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

2. **Actions**: These define all possible actions that the agent can take from a given state. Each action moves the agent from one state to another within the search space.
3. **Transition Model**: This model describes the outcome of each action, showing how one state changes to another. It helps the agent predict the results of its actions.
4. **Goal Test**: This is a function that checks if the current state matches the desired goal state. If the goal is reached, the function returns a positive result, indicating success.
5. **Path Cost**: This function assigns a numerical value or "cost" to each path taken. It helps evaluate different paths, allowing the agent to find the least costly or optimal path to the goal.

Parameters	Informed Search	Uninformed Search
Known as	Also known as **Heuristic Search**	Also known as **Blind Search**
Using Knowledge	Uses **knowledge** for the search process	Does **not use knowledge** for the search process
Performance	Finds a solution **more quickly**	Finds solutions **slower** compared to informed search
Completion	**May or may not** be complete	**Always complete**
Cost Factor	**Low cost**	**High cost**
Time	Consumes **less time** due to quicker search	Consumes **moderate time** due to slower search
Examples of Algorithms	• **Greedy Search** • **A* Search** • **AO* Search** • **Hill Climbing Algorithm**	• **Depth First Search (DFS)** • **Breadth First Search (BFS)** • **Branch and Bound**

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

In time complexity, which Informed Search is better and why? [5]
Answer:

Informed Search Algorithm Comparison:

Informed search algorithms, also called **heuristic searches**, use problem-specific information to guide the search process and find solutions more efficiently. Choosing the best-informed search algorithm depends on the problem and the quality of the heuristic function used.

A* Search

- **How it works**: A* combines the cost-so-far (`g(n)`) and an estimate of the remaining cost (`h(n)`) to decide which path to take.
- **Time Complexity**: The time complexity depends on how good `h(n)` is. If `h(n)` is **admissible** (doesn't overestimate the cost) and **consistent** (follows the triangle inequality), A* will find the optimal solution.
- **Worst Case**: The time complexity is exponential, `O(b^d)`, where `b` is the branching factor (maximum number of possible moves) and `d` is the depth of the solution. However, with a well-designed heuristic, A* often performs closer to linear time, `O(b^h)`, where `h` is the effective search depth.

Greedy Best-First Search

- **How it works**: Greedy Best-First Search only uses `h(n)` to expand the node that seems closest to the goal based on the heuristic.
- **Time Complexity**: Generally, it has lower time complexity than A*, but it doesn't guarantee the best solution. Greedy Best-First can find a solution fast if `h(n)` leads it efficiently toward the goal, but it may find a less optimal solution if `h(n)` is inaccurate or overestimates the cost.

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

Which Algorithm is Better?

- *A Search**: Preferred if there's an admissible and consistent heuristic available, and finding the best solution is critical.
- **Greedy Best-First Search**: Often better if a quick solution is acceptable, even if it may not be optimal, as it can be faster when a strong heuristic is available.

Constraints: A constraint is a rule or condition that limits the possible values a variable can take. It must be satisfied for the solution to be valid.

Backtracking Search: This is a method for solving **Constraint Satisfaction Problems (CSPs)**. It involves trying different values for the variables one by one. If a value does not satisfy the constraints, the search "backtracks" to try a different value until a valid solution is found or all possibilities are explored.

Arc Consistency: This is a concept in **CSPs** to ensure that for every variable, the possible values are consistent with the constraints between related variables. In other words, it checks that for every pair of variables, the value of one variable should have a compatible value in the other variable's domain according to the constraint

Here are some real-world examples of **Constraint Satisfaction Problems (CSPs)**:

1. **Assignment Problems**: For example, assigning teachers to classes based on their expertise and availability, or assigning employees to specific shifts.
2. **Timetable Problems**: For instance, deciding when and where each class will be held in a school or university, ensuring there are no conflicts with rooms or instructors.
3. **Transportation Scheduling**: Assigning vehicles to specific routes and times, ensuring that buses or trains are scheduled efficiently and do not overlap.

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

4. **Factory Scheduling**: Deciding the production schedule in a factory, ensuring that each machine is used efficiently while meeting deadlines and resource constraints.

How many solutions are there for the map-coloring problem in the following figure? The possible colors are three (red, green, and blue), and no neighboring regions can have the same colors.

Solution:

The task is to color each region of the map using three colors: red, green, or blue, while ensuring that no neighboring regions have the same color. The domains for each region (variable) are {red, green, blue}.

1. ***Coloring the SA (South Australia) region:***
    - We start by coloring the SA region. Since no colors are restricted at this point, we can choose any of the 3 colors (red, green, or blue).
    - So, there are **3 possibilities** for the SA region.
2. ***Coloring the WA (Western Australia) region:***
    - After coloring SA, we move to WA. Since WA is neighboring SA, it cannot have the same color as SA. This means we have 2 remaining color choices for WA.
    - So, there are **2 possibilities** for the WA region.
3. ***Coloring the other regions:***
    - The remaining regions (NT, NSW, V) can be colored based on the constraints, but since there are no specific restrictions provided, we can choose any color for each of them.
    - For T (Tasmania), as it's not connected to any other region, it can be colored with any of the 3 colors.
    - Thus, T has **3 possibilities**.

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

4. ***Total number of solutions:***
    - For each of the 6 possibilities of coloring SA and WA, there are 3 possibilities for coloring T.
    - Therefore, the total number of possible solutions is:
      3 × 2 × 3 = 18 solutions.

Thus, there are ***18 possible solutions*** for coloring the map using three colors.

Example of a CSP Problem:

Consider the map coloring problem where the goal is to assign colors to 7 cities in such a way that no neighboring regions have the same color. The available colors are red, green, and blue.

Variables:

WA (Western Australia)	SA (South Australia
T (Tasmania)	
NT (Northern Territory)	V (Victoria)
Q (Queensland)	NSW (New South Wales)

Domains:

- Each variable (city) can take one of the following colors: {red, green, blue}.

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

Constraints:

- No neighboring regions (cities) can have the same color. For example:
    - WA ≠ NT , WA ≠ SA, NT ≠ Q, NSW ≠ Q, NSW ≠ V

Solutions:

- A solution is a complete and consistent assignment of colors to each region where all the constraints are satisfied.
- For example, one possible solution could be:

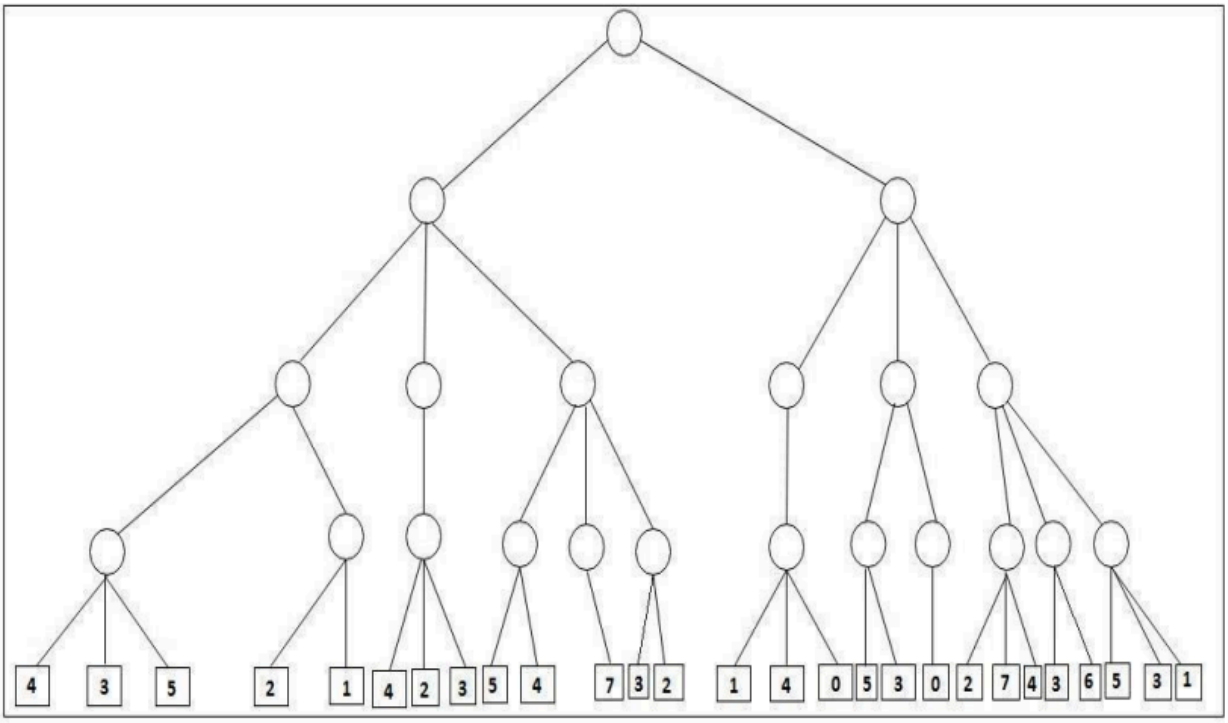
WA = red, NT = green, Q = red, NSW = green, V = red, SA = blue, T = green

In this solution, no two neighboring regions have the same color, and all variables are assigned a valid color from the domain.\

Definitions in the context of CSPs:

1. ***Domain:***
   The domain refers to the set of possible values that a variable can take in a CSP. It represents the allowable choices for that variable.
   Example: In a Sudoku puzzle, the domain for each cell is {1, 2, 3, 4, 5, 6, 7, 8, 9} because those are the possible numbers that can be placed in each cell.
2. ***Variable:***
   A variable in a CSP represents an unknown quantity that needs to be assigned a value from its domain to satisfy the constraints.
   Example: In a scheduling problem, variables could represent tasks, and each variable's domain would be the available time slots for the task.
3. ***Constraint:***
   A constraint defines the relationship or condition that must be satisfied by the values assigned to a set of variables.
   Example: In the N-Queens problem, the constraint is that no two queens can share the same row, column, or diagonal.

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)



What is the significance of the alpha and beta values in the alpha-beta pruning process, and how are they updated as the algorithm traverses the tree? (5)

The ***alpha*** *and* ***beta*** values in the alpha-beta pruning process are essential for efficiently pruning branches of the game tree while performing a Minimax search. They help determine which branches can be discarded without needing to fully explore them, making the search more efficient. Here's a breakdown of their significance and how they are updated during the search:

Alpha (α):

- ***Significance:***
  Alpha represents the best score found so far for the maximizing player

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

(e.g., Player A). It acts as a lower bound for the possible outcomes of a node for Player A.

- ***Initial Value:***
  It starts with a value of negative infinity because the maximizing player is looking for the highest possible score.
- ***Update:***
  As the algorithm explores the game tree, when evaluating a maximizing node (Player A's turn), the value of alpha is updated if the node's value is greater than the current alpha. This ensures that alpha always holds the highest score found on Player A's path so far.

Beta (β):

- ***Significance:***
  Beta represents the best score found so far for the minimizing player (e.g., Player B). It acts as an upper bound for the possible outcomes of a node for Player B.
- ***Initial Value:***
  It starts with a value of **positive infinity** because the minimizing player is looking for the lowest possible score.
- ***Update:***
  As the algorithm explores the tree, when evaluating a minimizing node (Player B's turn), the value of beta is updated if the node's value is less than the current beta. This ensures that beta always holds the lowest score found on Player B's path so far.

Updating Alpha and Beta During Traversal:

- ***Maximizing Node (Player A's turn):***
  If the value of the current node is greater than the current alpha, alpha is updated to the node’s value because it provides a better (higher) score for Player A.
- ***Minimizing Node (Player B's turn):***
  If the value of the current node is less than the current beta, **beta is**

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

updated to the node's value because it provides a better (lower) score for Player B.

Pruning Conditions:

- ***Pruning based on Alpha:***
  If the value of a node is greater than or equal to beta, then Player B (the minimizing player) would never choose this branch, as there is already a better option for Player B. In this case, the subtree rooted at this node can be pruned (not explored).
- ***Pruning based on Beta:***
  If the value of a node is **l**ess than or equal to alpha, then Player A (the maximizing player) would never choose this branch, as there is already a better option for Player A. In this case, the subtree rooted at this node can be pruned (not explored).

Summary:

- Alpha tracks the best (maximum) score for the maximizing player.
- Beta tracks the best (minimum) score for the minimizing player.
- As the algorithm explores nodes, alpha and beta are updated based on the values found.
- When a node violates the pruning conditions (either alpha ≥ beta or beta ≤ alpha), its subtree is discarded, reducing the search space and making the algorithm more efficient.

Max Node and Min Node in Minimax Algorithm:

In the Minimax algorithm:

- A **max node** represents a state where it's the maximizing player's **turn** (Player A). The goal of the maximizing player is to pick the move that **gives t**he highest possible score.

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

- A **min node** represents a state where it's the minimizing player's turn (Player B). The goal of the minimizing player is to pick the move that minimizes Player A's score.

These nodes are used to simulate the alternating turns of both players, helping the algorithm evaluate the best and worst outcomes for each player at every decision point in a game.

Alpha and Beta in Alpha-Beta Pruning:

In **alpha-beta pruning**:

- **Alpha (α)** is the best score found so far for the maximizing player. It starts at negative infinity and gets updated with higher scores as the tree is explored.
- Beta (β) is the best score found so far for the minimizing player. It starts at positive infinity and is updated with lower scores as the tree is explored.

These values help **prune** (eliminate) parts of the tree that can't influence the final decision:

- If a node's value is greater than or equal to beta, the minimizing player will avoid that branch.
- If a node's value is less than or equal to alpha, the maximizing player will avoid that branch.

Impact of Alpha-Beta Pruning on Efficiency:

Alpha-beta pruning *i*mproves the efficiency of the Minimax algorithm by reducing the number of nodes that need to be evaluated. It does this by pruning branches that are guaranteed to be suboptimal based on the alpha and beta values. As the algorithm explores the tree, it can stop exploring branches

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

early if they can't affect the outcome. This reduces the search space and speeds up the decision-making process without changing the final result.

Question 1: Check the validity of the following implications [Each Marks-2] P→ (Q→ R) equivalent to (P→ Q) → (P→R)

Answer:

P	Q	R	$Q \to R$	$(P \to Q)$	$(P \to R)$	$(P \to Q) \to (P \to R)$	$P \to (Q \to R)$
T	T	T	T	T	T	T	T
T	T	F	F	T	F	F	F
T	F	T	T	F	T	T	T
T	F	F	T	F	F	T	T
F	T	T	T	T	T	T	T
F	T	F	F	T	T	T	T
F	F	T	T	T	T	T	T
F	F	F	T	T	T	T	T

5th batch Solved.

1.a

Here is the PEAS description for a Virtual English Tutor Agent in a table format:

Parameter	Description
Performance	Accuracy of responses, Student engagement, Learning progress, Quality of feedback
Environment	Student inputs (text, speech), Course materials (lessons, quizzes), Student behavior, External resources (dictionaries, grammar checkers)
Actuators	Text and voice output, Text and voice input, Visual feedback (error highlighting)
Sensors	Speech recognition, Text input analysis, Emotion detection (engagement level), Progress tracking (performance data)

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

1 b. Define Agent and Rational Agent through real-time example [3]

- An agent is a software program that operates within an environment, perceives its surroundings, makes decisions, and takes actions to achieve specific goals. **Agent = Architecture + Program**
- ***Real-Time Example****:*
  A self-driving car is a perfect example of an agent. It uses sensors like cameras and radar to perceive its surroundings, processes the data, and takes actions accordingly. The car's goal is to navigate safely and efficiently to its destination. It detects obstacles, traffic signs, and other vehicles, and uses decision-making algorithms to decide when to accelerate, brake, or change lanes. The car interacts with the environment and makes intelligent decisions, making it an agent.

- ***Rational Agent****:*
  A rational agent always chooses the best course of action based on its knowledge and goals, aiming to maximize its expected outcome or utility.

- ***Real-Time Example****:*
  A virtual personal assistant, like Amazon's Alexa or Apple's Siri, is a rational agent. It uses natural language processing to understand user commands, fetches relevant information, and provides responses or performs tasks. These assistants learn from user interactions and feedback to improve their performance, always striving to offer accurate and useful responses. Their goal is to assist users effectively by making rational decisions on how to interpret and respond to queries.

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

1.c Distinguish between the following properties of a task environment:[3] i. Static or dynamic ii. Discrete or continuous iii. Single or multi-agent

i. ***Static or Dynamic:***

- ***Static****:* The environment does not change while the agent is performing its task. Example: Chess, where the state doesn't change until a move is made.
- ***Dynamic****:* The environment changes while the agent is acting. Example: A self-driving car, where traffic and road conditions constantly change.

ii. Discrete or Continuous:

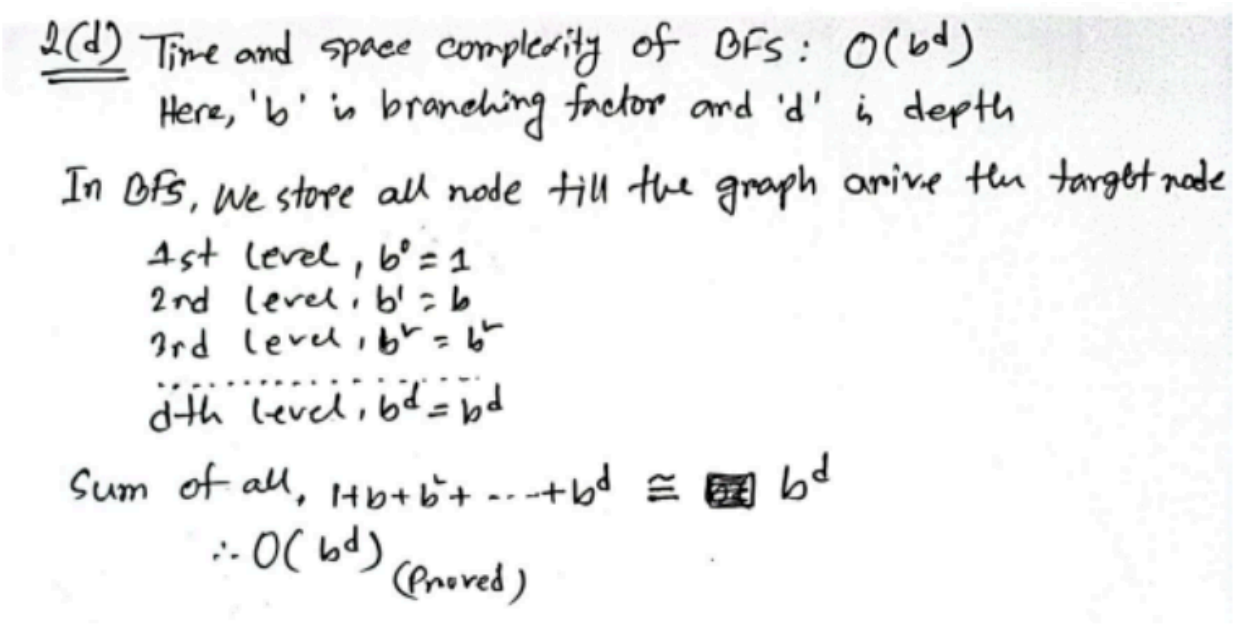
- ***Discrete****:* The environment has clear, distinct steps or states. Example: Tic-tac-toe, where moves are limited and clear.
- ***Continuous****:* The environment has an infinite number of states or actions. Example: Driving a car, where actions like speed and direction are continuously adjusted.

iii. ***Single or Multi-Agent:***

- ***Single-Agent****:* Only one agent is working to achieve its goal. Example: A robot cleaning a room.
- ***Multi-Agent****:* Multiple agents interact with each other. Example: Football, where players work or compete to achieve their goals.

2.c Prove that the time and space complexity of BFS is O(b^d). And explain it with an example [4]

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)



d) Does artificial intelligence provide an alternative to biological substances? [2]

No, artificial intelligence (AI) does not provide an alternative to biological substances. AI refers to computer systems designed to perform tasks requiring human-like intelligence, such as problem-solving and decision-making. However, it lacks the biological characteristics of living organisms, such as growth, reproduction, and metabolism. AI operates based on algorithms and data processing, while biological substances are made up of organic molecules and possess life functions. AI can assist and enhance human tasks but does not replace biological life.

3.a) Prove that A search is optimal if h(n) is admissible(Grrohon Joggo).[3]*

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

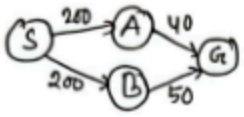
⇒ In A* search ⇒ $f(n) = g(n) + h(n)$

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

Overstimation : $h(n) \geq h^*(n)$

Here, $h(n)$ → estimated heuristic value

$h^*(n)$ → Actual value



$g(A) = 200$
$g(B) = 200$
$h^*(A) = 40$
$h^*(B) = 50$

Case-1: (Overestimate)

$h(A) = 80$
$h(B) = 70$] $> h^*$

$f(A) = g(A) + h(A)$
$= 200 + 80$
$= 280$

$f(B) = g(B) + h(B)$
$= 200 + 70$
$= 270$

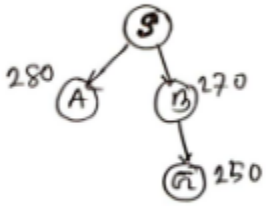
Case-2: (Underestimate)

$h(A) = 30$
$h(B) = 20$] $< h^*$

$f(A) = g(A) + h(A)$
$= 200 + 30$
$= 230$

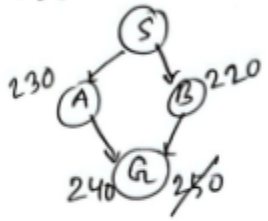
$f(B) = g(B) + h(B)$
$= 200 + 20$
$= 220$

$f(G) = g(G) + h(G)$
$= 250 + 0$
$= 250$



Here, go to G through B and $f(G) = 250$ and $f(B) = 270$ so, don't check to $f(A)$

$f(G) = g(G) + h(G)$
$= 250 + 0$
$= 250$



Here, go to G through B and $f(G)$ is greater than $g(B)$, so, now check $g(A)$ ∴ $f(G) = 240 + 0 = 240$ so, here we got optimal value

6.b) What is natural language processing (NLP)? Describe the development steps for a typical NLP application.[3]

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

Natural Language Processing (NLP) is a branch of artificial intelligence that helps computers understand, interpret, and generate human language. It includes tasks like speech recognition, translation, and text analysis.

Steps to Develop a Typical NLP Application:

1. ***Data Collection and Preprocessing****:*
    - Collect text or speech data (e.g., articles, books, or social media posts).
    - Clean and process the data by removing unnecessary words and characters. This includes breaking the text into smaller parts (tokenization) and removing common words (like "the" or "is").
2. ***Text Representation****:*
    - Convert the text into a format the computer can understand, like numbers or vectors (e.g., Word2Vec or GloVe embeddings).
3. ***Modeling and Training****:*
    - Use machine learning or deep learning models to learn from the data. The model identifies patterns and meanings to perform tasks (like sentiment analysis or translation).
4. ***Evaluation and Fine-Tuning****:*
    - Test the model's performance using metrics (accuracy, precision, etc.). Adjust the model if needed to improve results.
5. ***Deployment****:*
    - Deploy the trained model in a real-world application, such as a chatbot or translation tool, for users to interact with.

This process ensures the NLP application works well and provides useful results based on human language.

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

Aspect	Strong AI	Weak AI
Definition	AI with human-like intelligence and consciousness	AI focused on specific tasks without consciousness
Capabilities	Can understand, learn, and reason across various domains	Limited to predefined tasks; lacks broad reasoning
Example Applications	Hypothetical; true human-like robots	Virtual assistants, autonomous cars, chatbots
Goal	Achieve general intelligence similar to humans	Enhance specific task efficiency and accuracy

Among given Option which Algorithm Requires Less Memory? And Why? Give With An Example. 1.Optimal Search 2.Depth Search 3.Breadth First search 4. Linear search.

1. ***Optimal Search:***

- **Memory Requirement**: Varies depending on the specific optimal search algorithm being used. For example, **Binary Search** (an optimal search in sorted arrays) has low memory requirements since it uses only constant O(1)O(1)O(1) space.
- **Example**: Binary Search only keeps track of the search bounds (left and right indices), using minimal memory.
- **Conclusion**: Very efficient in memory usage, as it doesn't need to store additional nodes or paths.

2. ***Depth-First Search (DFS):***

- **Memory Requirement**: Uses memory proportional to the maximum depth of the tree or graph (up to O(d)O(d)O(d), where ddd is the depth).
- **Example**: In a graph with deep paths, DFS only needs to remember nodes along the current path, making it efficient for deep structures with few branches.

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

- **Conclusion**: DFS generally uses less memory than BFS in large, deep graphs, as it only needs to store nodes on the current path.

3. ***Breadth-First Search (BFS):***

- **Memory Requirement**: Requires memory proportional to the width of the tree or graph level, which can be up to O(bd)O(b^d)O(bd), where bbb is the branching factor and ddd is the depth.
- **Example**: In a binary tree, BFS stores all nodes at the current level, which can grow quickly for wide trees.
- **Conclusion**: BFS tends to require more memory than DFS, especially in wide trees or graphs.

4. ***Linear Search:***

- **Memory Requirement**: Constant O(1)O(1)O(1), as it simply traverses each element in a list sequentially without additional storage.
- **Example**: Scanning an array from start to finish to find a target value.
- **Conclusion**: Very memory-efficient, as it only requires a single pointer or index variable, regardless of the list size.

Summary:

In terms of memory efficiency (from lowest to highest memory requirement):

1. **Linear Search** – requires constant memory.
2. **Optimal Search** (like Binary Search) – also very efficient, with minimal memory usage.
3. **DFS** – more memory-efficient than BFS in deep trees or graphs.
4. **BFS** – requires more memory in wide or large trees due to storing nodes at each level.

So, **Linear Search** and **Optimal Search** use the least memory, followed by **DFS**, while **BFS** requires the most memory for wide structures.

First order Logic Questions: FOL

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

In time complexity, which Uninformed Search is better and why? [5]

Time Complexity of Uninformed Search Algorithms:

1. ***Depth-First Search (DFS)**:*
    - **Time Complexity**: O(b^m), where `b` is the branching factor and `m` is the maximum depth.
    - **Best For**: Shallow goal nodes and memory efficiency. DFS explores one branch deeply before backtracking, requiring less memory than BFS.
    - **Cons**: May explore deep branches unnecessarily, leading to poor performance if the goal is deep.
2. ***Breadth-First Search (BFS)**:*
    - **Time Complexity**: O(b^d), where `d` is the depth of the shallowest solution.
    - **Best For**: Finding the shortest path when all actions have equal cost. Guarantees an optimal solution when the goal is at a shallow depth.
    - **Cons**: High memory usage, storing all nodes at each depth level.
3. ***Uniform Cost Search (UCS)**:*
    - **Time Complexity**: O(b^d) in the worst case, similar to BFS.
    - **Best For**: Problems with variable path costs. It ensures the optimal solution in terms of path cost.
    - **Cons**: Can be slower than BFS if path costs vary significantly.

Conclusion:

- **DFS** is optimal for shallow goals with memory constraints.
- **BFS** guarantees the shortest path in terms of steps for shallow goals.
- **UCS** is best for problems where path costs vary and you need the optimal path based on cost.

The choice depends on the problem's characteristics (goal depth, branching factor, and action cost).

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

Constraint Satisfaction Problems (CSPs) are a fundamental concept in Artificial Intelligence (AI) used to solve problems where the goal is to find a solution that satisfies a set of constraints. CSPs are common in various applications such as scheduling, planning, Map Coloring, Resource allocation, and games like Sudoku. In AI, CSPs provide a structured way to model problems where each possible solution must meet specific restrictions or rules.

Video

111

Key Terms in CSPs:

Variables

- **Definition**: Variables are the entities in the problem that need to be assigned values.
- **Example**: In map coloring, each region on the map is a variable. For instance, on a map with regions A, B, C, D, and E, each of these regions represents a different variable that needs to be assigned a color.

Domains

- **Definition**: Each variable has a domain, which is the set of possible values it can take.
- **Example**: For map coloring, if we are using three colors — Red, Green, and Blue — then the domain for each region (variable) is {Red, Green, Blue}. This means each region can be assigned one of these colors.

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

Constraints

- **Definition**: Constraints specify the relationships and restrictions between variables. They limit the combinations of values that variables can simultaneously take.

- **Example**: In map coloring, a constraint is that no two neighboring regions should share the same color. So, if region A is adjacent to region B, they cannot both be colored Red. This constraint applies to all pairs of neighboring regions.

Goal

- **Definition**: The objective of a CSP is to find an assignment of values to variables such that all constraints are satisfied.
- **Example**: In map coloring, the goal is to assign a color to each region so that no two neighboring regions have the same color. This ensures that all constraints are met, and the map is properly colored.

Explanation of Forward Checking, Backward Checking and Arc Consistency:

Forward Checking, Backward Checking, and Arc Consistency (AC) are all techniques used in solving Constraint Satisfaction Problems (CSPs) to make the search process more efficient by reducing the solution space. Here's a breakdown of each:

1. Forward Checking

Forward Checking is a method used to prevent future conflicts as we assign values to variables during the CSP search. When a value is assigned to a variable, Forward Checking immediately checks if the assignment affects the domain of the neighboring (or "unassigned") variables.

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

- **How it Works**: After assigning a value to a variable, Forward Checking looks at the constraints involving that variable and other unassigned variables. It removes values from the domains of unassigned variables that would lead to constraint violations.
- **Example**: In Sudoku, if a cell is assigned the value "5," Forward Checking would remove "5" from the domain of all cells in the same row, column, and block, as "5" would be an invalid choice for those cells.
- **Pros and Cons**: Forward Checking reduces the number of possible future choices early on, potentially avoiding dead ends. However, it doesn't eliminate all possible inconsistencies, as it only considers direct neighbors.

2. Backward Checking (Backtracking)

Backward Checking, commonly known as Backtracking, is a general approach in CSPs where we assign values to variables one by one and backtrack whenever a constraint violation is detected.

- **How it Works**: In Backtracking, values are assigned sequentially. If a partial assignment results in a constraint violation, Backtracking undoes (or "backs up" from) the most recent assignment and tries a new value. This process continues until a solution is found or all possibilities are exhausted.
- **Example**: If you are assigning colors to regions on a map (a classic CSP problem), and assigning "Red" to one region prevents any valid color assignment for neighboring regions, Backtracking would undo that assignment and try a different color.
- **Pros and Cons**: While straightforward and systematic, pure Backtracking can be slow in large problems as it explores each possibility without additional checks. However, combined with other techniques like Forward Checking, it becomes much more efficient.

3. Arc Consistency (AC)

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

Arc Consistency is a more advanced technique that enforces consistency between pairs of variables, helping to eliminate invalid values from domains early in the search process.

- **How it Works**: Arc Consistency ensures that, for every pair of variables involved in a constraint (referred to as an "arc"), each value in the domain of one variable has a valid corresponding value in the domain of the other. If no such value exists, that value is removed from the domain.
    - The **AC-3 Algorithm** is commonly used to enforce arc consistency. It repeatedly checks all arcs in the problem until no more values can be removed from any domain.
- **Example**: In a map-coloring problem, if one region has a limited set of colors due to neighboring assignments, Arc Consistency would remove colors from neighboring regions' domains that would lead to conflicts, thus narrowing down the choices.
- **Pros and Cons**: Arc Consistency can greatly reduce the search space by pruning invalid values early on. However, maintaining arc consistency can be computationally intensive, as it may need to check and revise domains repeatedly.

Comparing Forward Checking, Backward Checking, and Arc Consistency

- **Forward Checking** is useful for detecting conflicts at each step as values are assigned, preventing some invalid choices from progressing further.
- **Backward Checking (Backtracking)** is the base search method and can be improved when used with Forward Checking and heuristics.

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

- **Arc Consistency** takes a more global view by refining the domains before any specific assignment is made, which can prevent large portions of the search space from needing to be explored.

Which is Better?

- **For small or simple CSPs** (like the map coloring of a few regions), **Backtracking or Forward Checking** may be enough because the problem size doesn't justify the overhead of full arc consistency.
- **For larger or more complex CSPs** with many constraints, **Arc Consistency** (AC-3) is usually more efficient in the long run because it prunes the search space extensively, reducing the need for backtracking.

In summary:

- **Arc Consistency > Forward Checking > Backtracking** for complex problems due to its ability to reduce the search space significantly.
- For simpler CSPs, **Forward Checking** strikes a good balance between efficiency and ease of implementation.

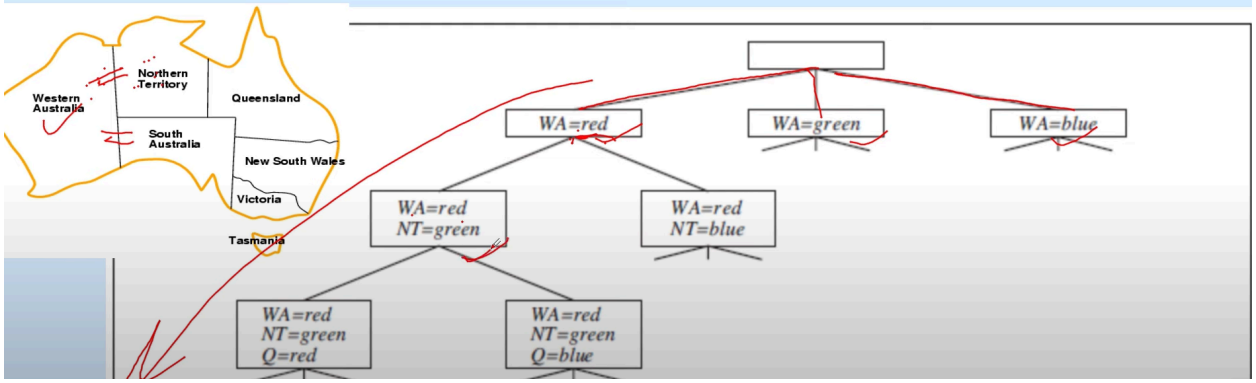
AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

Slide from video:

2) Arc consistency: A variable in a CSP is **arc-consistent** if every value in its domain satisfies the variable's binary constraints.

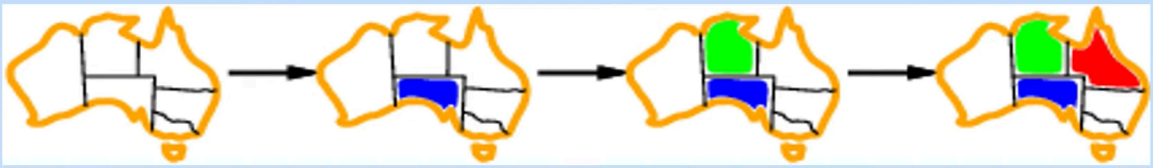
- More formally, Xi is arc-consistent with respect to another variable Xj if for every value in the current domain Di there is some value in the domain Dj that satisfies the binary constraint on the arc (Xi, Xj).
- A network is arc-consistent if **every variable is arc consistent with every other variable.**
- For example, consider the constraint Y = X^2 where the domain of both X and Y is the set of positive integers below 10. We can write this constraint explicitly as (X, Y), {(0, 0), (1, 1), (2, 4), (3, 9))} .
- To make X arc-consistent with respect to Y , we reduce X's domain to {0, 1, 2, 3}. If we also make Y arc-consistent with respect to X, then Y 's domain becomes {0, 1, 4, 9} and the whole CSP is arc-consistent.

- **Depth-first search for CSPs with single-variable assignments is called backtracking search.**
- In backtracking, only one successor is generated at a time rather than all successors; each partially expanded node remembers which successor to generate next. In this way, only O(m) memory is needed rather than O(bm).



AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

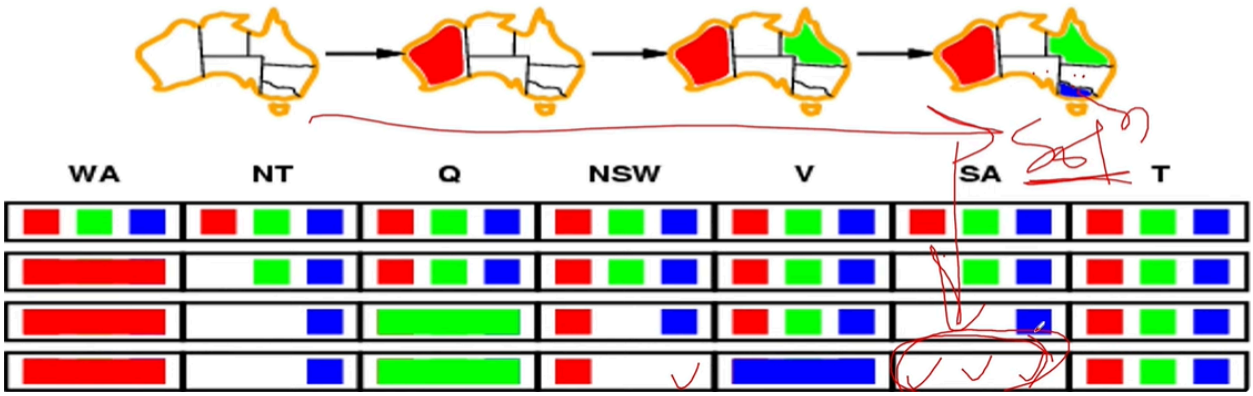
- Most constrained variable:
  choose the variable with the fewest legal values



- minimum remaining values (MRV) heuristic

- Tie-breaker among most constrained variables
- Most constraining variable:
  - choose the variable with the most constraints on remaining variables

- Idea:
  - Keep track of remaining legal values for unassigned variables
  - Terminate search when any variable has no legal values



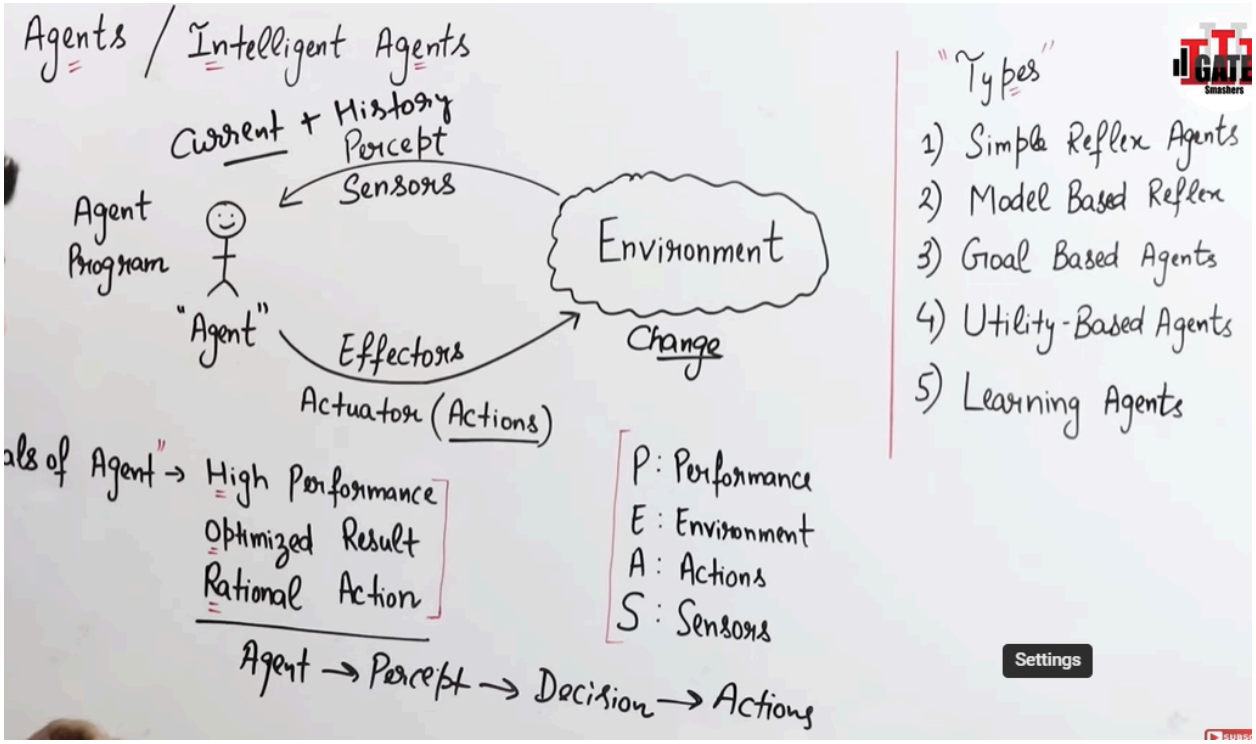
Procedure (Uniform Cost Search)

1. Initialize the priority queue with the start node `S` and cost `0`.
2. Push the path with the lowest cost.
3. Expand this node to all its neighbors, calculate the path cost, and push unvisited or lower-cost paths to the priority queue.
4. Repeat until the goal node `G` is dequeued with the minimum cost.

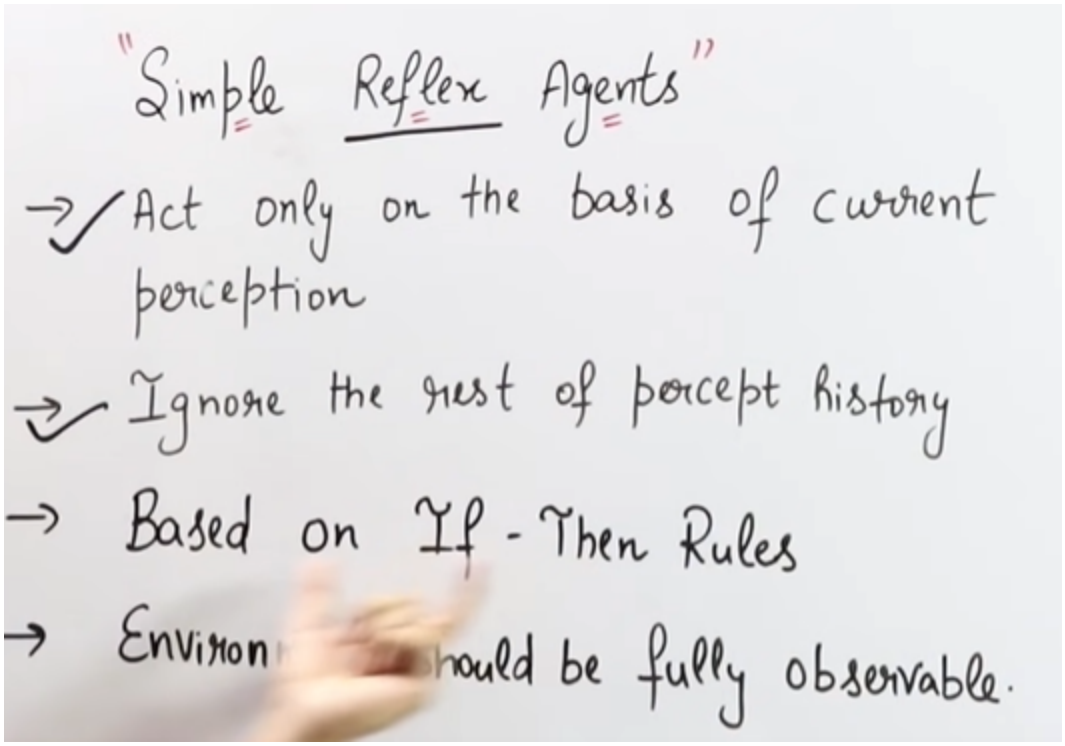
AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

*Procedure (A Algorithm)**

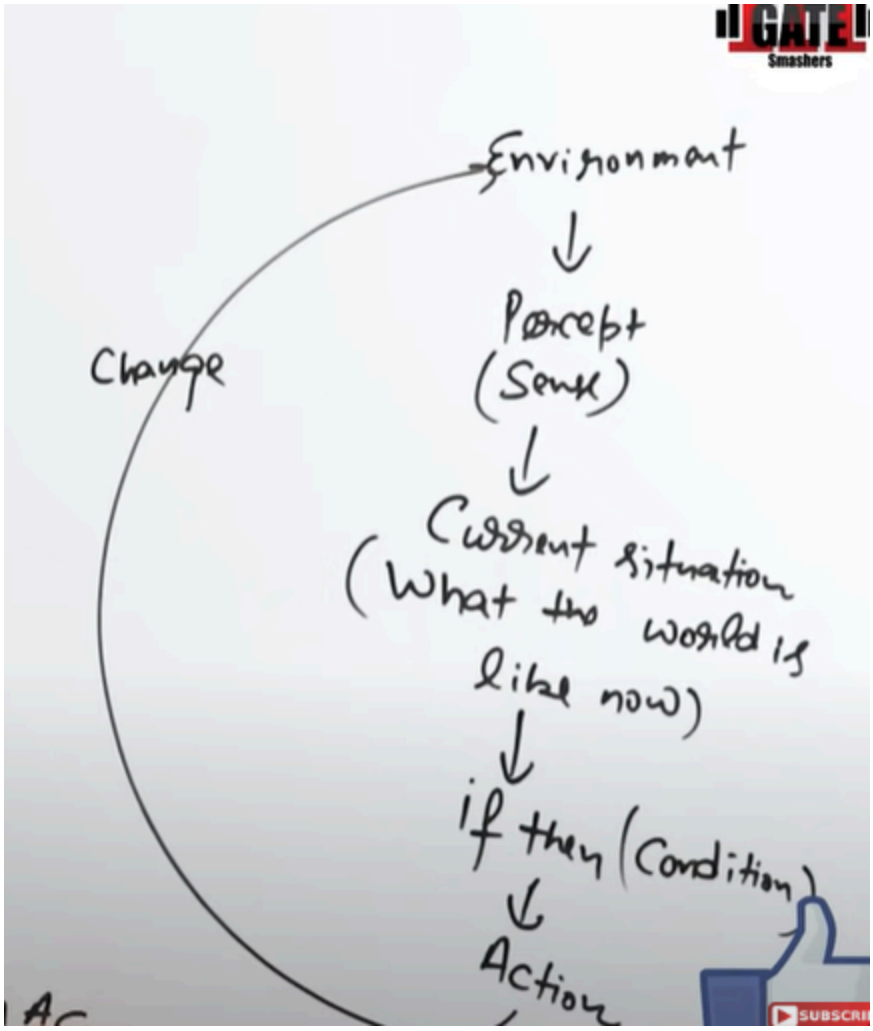
1. Initialize the priority queue with start node S and cost 0 (f(n) = g(n) + h(n)).
2. **Pop** the node with the lowest f(n).
3. Check if the **goal** node G is reached. If yes, return the path and cost.
4. **Expand** the current node's neighbors, calculate g(neighbor) and f(neighbor), and push unvisited or lower-cost paths to the priority queue.
5. **Repeat** until the goal is found or the queue is empty.



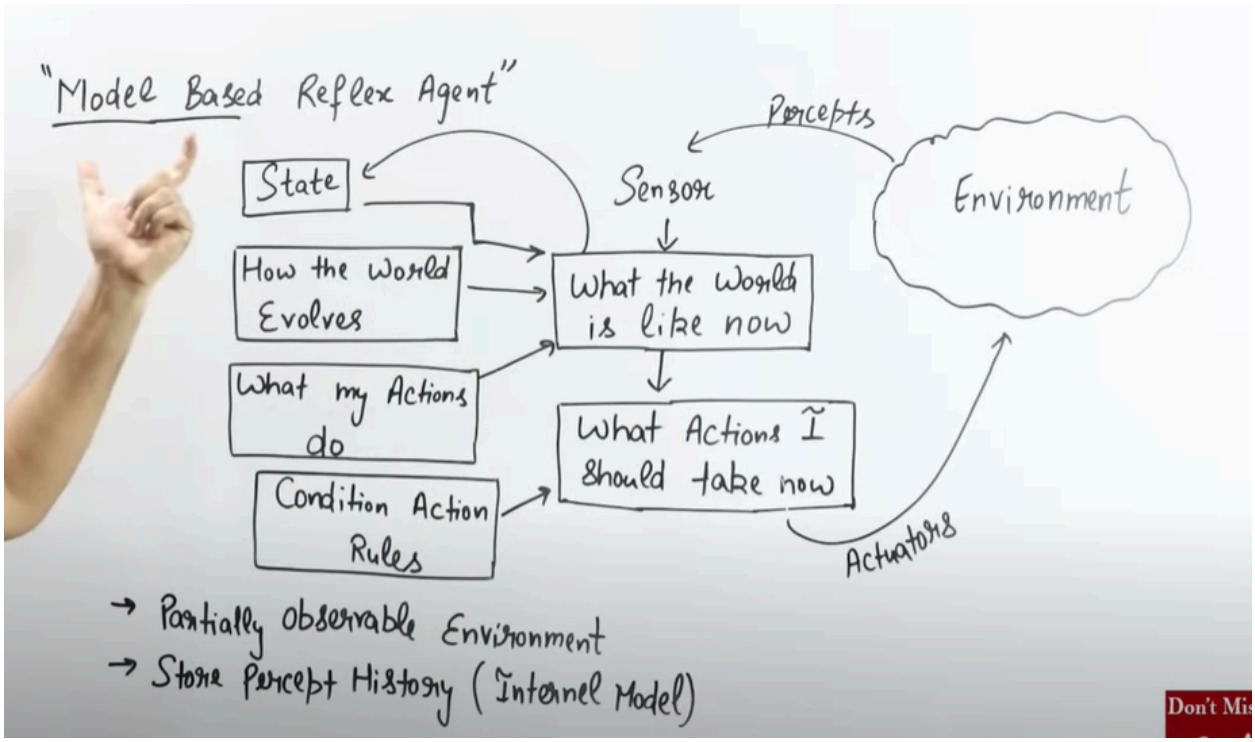
AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)



AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)



AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)



Step 2: Create the Truth Tables for Each Statement

We'll make a truth table for both statements and check if they are true in all cases, which would establish their validity.

Statement 1: $P \to (Q \to R) \equiv (P \to Q) \to (P \to R)$

Let's start with a truth table to analyze each part of this equivalence.

P	Q	R	$Q \to R$	$P \to (Q \to R)$	$P \to Q$	$P \to R$	$(P \to Q) \to (P \to R)$	Equivalent?
T	T	T	T	T	T	T	T	T
T	T	F	F	F	T	F	F	T
T	F	T	T	T	F	T	T	T
T	F	F	T	T	F	F	T	T
F	T	T	T	T	T	T	T	T
F	T	F	F	T	T	T	T	T
F	F	T	T	T	T	T	T	T
F	F	F	T	T	T	T	T	T

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

In the table:

- **Column 5** represents $P \to (Q \to R)$.
- **Column 8** represents $(P \to Q) \to (P \to R)$.

The two columns match in all cases, meaning **Statement 1 is valid** (the expressions are equivalent).

Statement 2: $[(P \to Q) \vee (R \to S)] \to [(P \vee R) \to (Q \vee S)]$

Now, let's construct the truth table for this statement.

P	Q	R	S	$P \to Q$	$R \to S$	$(P \to Q) \vee (R \to S)$	$P \vee R$	$Q \vee S$	$(P \vee R) \to (Q \vee S)$	Equivalent?
T	T	T	T	T	T	T	T	T	T	T
T	T	T	F	T	F	T	T	T	T	T
T	T	F	T	T	T	T	T	T	T	T
T	T	F	F	T	T	T	T	T	T	T
T	F	T	T	F	T	T	T	T	T	T
T	F	T	F	F	F	F	T	F	F	T
T	F	F	T	F	T	T	T	T	T	T
T	F	F	F	F	T	T	T	F	F	T
F	T	T	T	T	T	T	T	T	T	T
F	T	T	F	T	F	T	T	T	T	T
F	T	F	T	T	T	T	F	T	T	T
F	T	F	F	T	T	T	F	T	T	T
F	F	T	T	T	T	T	F	T	T	T
F	F	T	F	T	F	T	F	F	T	T
F	F	F	T	T	T	T	F	T	T	T
F	F	F	F	T	T	T	F	F	T	T

In this table:

- **Column 7** represents $(P \to Q) \vee (R \to S)$.
- **Column 10** represents $(P \vee R) \to (Q \vee S)$.

Since the implication holds in every row (Column 10 is always true), **Statement 2 is also valid**.

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

1. **Implication (→):**
   - Symbol: $A \rightarrow B$
   - Definition: $A \rightarrow B$ is false only when A is true and B is false. In all other cases, it is true.
   - Truth table:

A	B	$A \rightarrow B$
T	T	T
T	F	F
F	T	T
F	F	T

2. **Or (Disjunction, ∨):**
   - Symbol: $A \vee B$
   - Definition: $A \vee B$ is true if at least one of A or B is true.
   - Truth table:

A	B	$A \vee B$
T	T	T
T	F	T
F	T	T
F	F	F

3. **And (Conjunction, ∧):**
   - Symbol: $A \wedge B$
   - Definition: $A \wedge B$ is true if both A and B are true.
   - Truth table:

A	B	$A \wedge B$
T	T	T
T	F	F
F	T	F
F	F	F

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

4. **Negation (Not, ¬)**:
   - Symbol: $\neg A$
   - Definition: $\neg A$ is true if A is false, and false if A is true.
   - Truth table:

A	$\neg A$
T	F
F	T

5. **Biconditional (↔)**:
   - Symbol: $A \leftrightarrow B$
   - Definition: $A \leftrightarrow B$ is true if A and B are both true or both false (they have the same truth value).
   - Truth table:

A	B	$A \leftrightarrow B$
T	T	T
T	F	F
F	T	F
F	F	T

6. **Exclusive Or (XOR, ⊕)**:
   - Symbol: $A \oplus B$
   - Definition: $A \oplus B$ is true if either A or B is true, but not both.
   - Truth table:

A	B	$A \oplus B$
T	T	F
T	F	T
F	T	T
F	F	F

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

7. **Nand (¬∧)**:
    - Symbol: $A \uparrow B$
    - Definition: $A \uparrow B$ is true if $A \wedge B$ is false (the negation of "and").
    - Truth table:

A	B	$A \uparrow B$
T	T	F
T	F	T
F	T	T
F	F	T

8. **Nor (¬∨)**:
    - Symbol: $A \downarrow B$
    - Definition: $A \downarrow B$ is true if $A \vee B$ is false (the negation of "or").
    - Truth table:

A	B	$A \downarrow B$
T	T	F
T	F	F
F	T	F
F	F	T

Local Search Algorithms

Local search algorithms are well-suited for solving optimization problems where the main goal is to find an optimal or near-optimal solution based on an objective function. These problems are distinct in that:

- There is no specific start state, and the path to a solution is not relevant.
- Instead, the focus is on evaluating potential solutions based on an objective function that quantifies solution quality. The aim is to either

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

minimize or maximize the value of this function to obtain the best possible solution.

Examples:

1. **Traveling Salesman Problem (TSP)**
    - **Objective:** Find the shortest possible route that visits each city once and returns to the starting point.
    - **State Space:** All possible routes (tours) connecting the set of cities.
    - **Objective Function:** Total length of the tour, which the algorithm seeks to minimize.
2. **n-Queens Problem**
    - **Objective:** Place nnn queens on an n×nn \times nn×n chessboard so that no two queens threaten each other.
    - **State Space:** All possible configurations of n queens on the board.
    - **Objective Function:** Number of pairwise conflicts between queens (to be minimized).

Local search algorithms are especially useful for large state spaces where traditional search methods are impractical.

Forward Chaining

Forward chaining starts with known facts and uses rules to find new facts step-by-step until it reaches a conclusion. This is called a *data-driven* approach because it moves forward based on what we already know.

Example: Rules:

- If an animal has feathers, it is a bird.
- If an animal is a bird and cannot fly, it is a penguin.

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

Facts:

- The animal has feathers.
- The animal cannot fly.

Starting with the facts, we apply the rules: since the animal has feathers, it's a bird. Then, since it's a bird and cannot fly, we conclude it's a penguin.

Backward Chaining

Backward chaining starts with a goal and works backward, using rules to see if the known facts support it. This is a *goal-driven* approach.

Example: Goal: Is the animal a penguin?

1. To confirm, we need to show it's a bird and cannot fly.
2. From the known facts, we see the animal has feathers, so it's a bird, and it cannot fly.

By working backward, we confirm that the animal is a penguin.

6th batch(8):

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

a.

1. **No one talks**: $\forall x \, (M(x) \rightarrow \neg T(x))$
2. **Everyone loves himself**: $\forall x \, L(x, x)$
3. **Everyone loves everyone**: $\forall x \, \forall y \, L(x, y)$
4. **Everyone loves everyone except himself**: $\forall x \, \forall y \, (x \neq y \rightarrow L(x, y))$
5. **Not all cars have carburetors**: $\neg \forall x \, (C(x) \rightarrow H(x))$
6. **Every connected and circuit-free graph is a tree**: $\forall x \, ((C(x) \wedge F(x)) \rightarrow T(x))$
7. **All that glitters is not gold**: $\exists x \, (G(x) \wedge \neg Go(x))$
8. **Not all that glitters is gold**: $\neg \forall x \, (G(x) \rightarrow Go(x))$
9. **There is a barber who shaves all men in the town who do not shave themselves**: $\exists x \, (B(x) \wedge \forall y \, ((M(y) \wedge \neg S(y, y)) \rightarrow S(x, y)))$

b.

Limitations of First-Order Logic (FOL):

1. **Cannot Express Beliefs or Knowledge**: FOL lacks the ability to represent statements about knowledge or beliefs, like "Alice knows Bob is honest."
2. **No Probabilistic Reasoning**: FOL cannot handle uncertain or probabilistic information, as it only expresses absolute truths (true or false).

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

C.

To prove that the proposition $A \wedge \neg A \vee (B \vee C)$ is a contradiction, we can use a truth table. A contradiction is a statement that is always false, regardless of the truth values of its variables.

Proposition:

$A \wedge \neg A \vee (B \vee C)$

Steps:

1. **List all possible truth values** for the variables A, B, and C.
2. **Evaluate** the components of the proposition step by step:
   - A
   - $\neg A$
   - $B \vee C$
   - $A \wedge \neg A$
   - Final expression $A \wedge \neg A \vee (B \vee C)$

Truth Table:

A	B	C	$\neg A$	$B \vee C$	$A \wedge \neg A$	$A \wedge \neg A \vee (B \vee C)$
T	T	T	F	T	F	T
T	T	F	F	T	F	T
T	F	T	F	T	F	T
T	F	F	F	F	F	F
F	T	T	T	T	F	T
F	T	F	T	T	F	T
F	F	T	T	T	F	T
F	F	F	T	F	F	F

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

Conclusion:

The expression $A \land \neg A \lor (B \lor C)$ is not a contradiction. It evaluates to true for some combinations of truth values for A, B, and C. Therefore, it is not always false. Thus, the proposition is **not a contradiction**.

d

Truth Table:

A	B	$A \lor B$	$\neg A$	$\neg B$	$\neg A \land \neg B$
T	T	T	F	F	F
T	F	T	F	T	F
F	T	T	T	F	F
F	F	F	T	T	T

Conclusion:

The truth table shows that $(A \lor B)$ and $(\neg A \land \neg B)$ **do not have the same truth values** in every possible case. Therefore, **the two propositions are not equivalent**.

In fact, they are **contradictory**. The statement $(A \lor B)$ is the opposite of $(\neg A \land \neg B)$, which is a form of De Morgan's Law, where:

- $(A \lor B)$ is true when at least one of A or B is true.
- $(\neg A \land \neg B)$ is true only when both A and B are false.

Thus, these propositions are not logically equivalent.

7.a

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

What is the Probabilistic resoning? how is probilistic Reasoning different from logical Reasoning ?

Ans:

Probabilistic reasoning involves making inferences and decisions based on the likelihood or probability of events happening, rather than definite true/false values. It is used to model uncertainty and deal with situations where information is incomplete or uncertain

or example, if you know there is a 70% chance of rain tomorrow, probabilistic reasoning allows you to make decisions based on this uncertainty, like carrying an umbrella.

Difference Between Probabilistic and Logical Reasoning:

Aspect	Probabilistic Reasoning	Logical Reasoning
Inference	Based on probabilities, dealing with uncertainty.	Based on certainty, true or false.
Handling Uncertainty	Handles uncertainty and partial knowledge.	Assumes complete and certain information.
Conclusion	Conclusions are likely, not guaranteed.	Conclusions are definite (true/false).

Key Difference:

- **Probabilistic Reasoning** involves uncertainty and likelihood, while **Logical Reasoning** assumes certainty and clear truth values.

agee Video pre Solution.

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

7.b (i)

$P(C, R, \sim S, W)$.

$= P(W | \sim S, R) \times P(\sim S | C) \times P(R | C) \times P(C)$

$= 0.90 \times 0.90 \times 0.8 \times 0.5$ → with the help of Table.

$= \boxed{0.34}$.

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

7.b (ii)

Ans.

~~P(A)~~ $P(A|B) = \frac{P(A \wedge B)}{P(B)}$

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

↳ B, যেটি A ঘটার সম্ভাবনা

Let, cloudy → C

Not cloudy → $\sim C$.

Raining → R.

Not raining → $\sim R$.

Sprinkler is on → S. grass is wet → W.

$P(\sim C, \sim R, S, W)$

$= P(W|S,R) * P(S|\sim C) * P(\sim R|\sim C) * P(\sim C)$

$= 0.90 \times 0.50 * 0.8 * 0.5$

$= 0.18.$

AVAILABLE AT:
Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)