The Yenepoya Institute of Arts, Science, Commerce and Management
A Constituent Unit of Yenepoya (Deemed to be University)
Introduction to Artificial Intelligence
QP Code: H024

Semester – End Examination Answer Key
SECTION – A

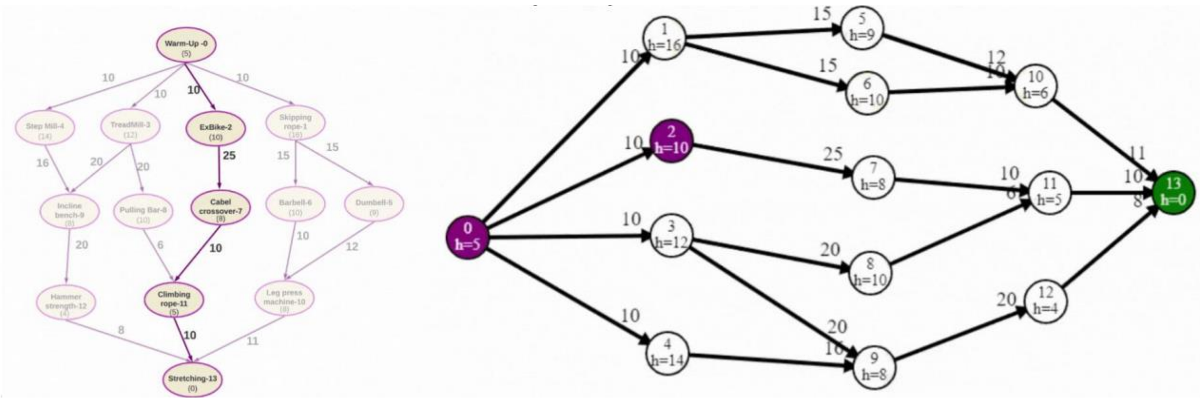
MULTIPLE CHOICE QUESTIONS **(15 X 1= 15)**

1. c. The simulation of human intelligence in machines
2. b. Internet of Things
3. c. Image and video analysis
4. b. To explore the search space and find a solution
5. b. By analyzing large datasets for threat detection
6. c. Knowledge Acquisition
7. d. p implies q
8. b. ¬Cat(x) ∨ Animal(x)
9. a. Start with facts vs Start with goals
10. a. Words
11. d. All of the above
12. c. To identify and classify entities like names, places, and organizations
13. a. It is more efficient than bottom-up parsing
14. c. Database
15. c. rules

SECTION – B

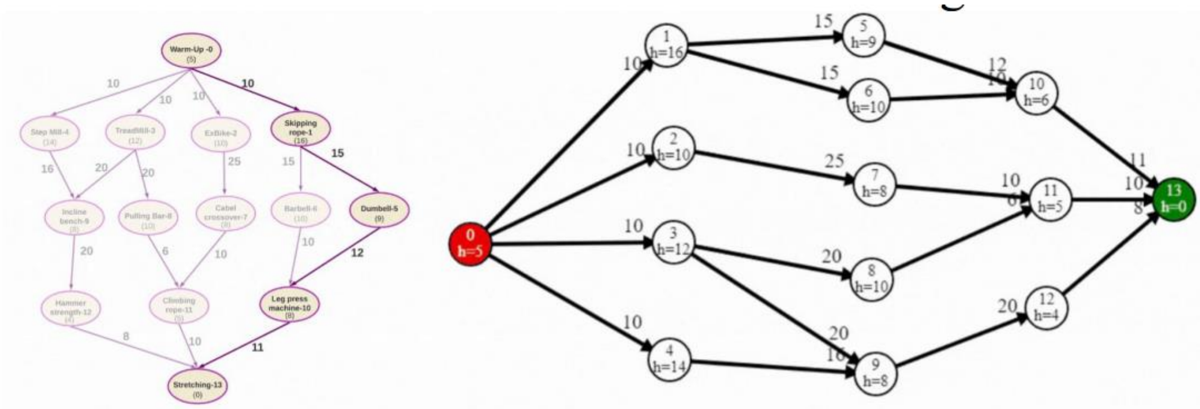
ANSWER ANY THREE QUESTIONS **(3 X 10= 30)**

16. What is Problem Solving in AI? Explain the steps involved in it.
Answer:
1. Heuristics
   - The heuristic method helps comprehend a problem and devises a solution based purely on experiments and trial and error methods.
   - However, these heuristics do not often provide the best optimal solution to a specific problem.
   - Instead, these undoubtedly offer efficient solutions to attain immediate goals.
   - Therefore, the developers utilize these when classic methods do not provide an efficient solution for the problem.
   - Since heuristics only provide time-efficient solutions and compromise accuracy, these are combined with optimization algorithms to improve efficiency.
   - A heuristic is a way which might not always be guaranteed for best solutions but guaranteed to find a good solution in reasonable time.
   - Informed search can solve much complex problem which could not be solved in another way.
   - Ex: Traveling salesman problem.



2. Uninformed/Blind Search

- It is a search algorithm that explores a problem space without any specific knowledge or information about the problem other than the initial state and the possible actions to take.
- It lacks domain-specific heuristics or prior knowledge about the problem.
- Uninformed search applies a way in which search tree is searched without any information, so it is also called blind search.
- It examines each node of the tree until it achieves the goal node.



Breadth-First Search (BFS)

- BFS is a graph traversal algorithm that systematically explores a graph level by level.
- It starts at a specific node and visits all its immediate neighbors before moving to the next level.

Depth-First Search (DFS)

- Depth-first search is a recursive algorithm for traversing a tree or graph data structure.
- It is called the depth-first search because it starts from the root node and follows each path to its greatest depth node before moving to the next path.
- DFS uses a stack data structure for its implementation.

17. Describe how AI is used in chess and real-time strategy games. What are the key AI techniques involved?

Answer:

AI in Chess and Real-Time Strategy (RTS) Games

AI in Chess

- Chess AI evaluates possible moves using search algorithms like Minimax and optimizes it with Alpha-Beta Pruning to reduce the search space.
- Heuristic Evaluation helps score positions based on factors like material advantage and king safety.
- Modern chess engines, such as AlphaZero, use Machine Learning (ML) and reinforcement learning to improve through self-play.

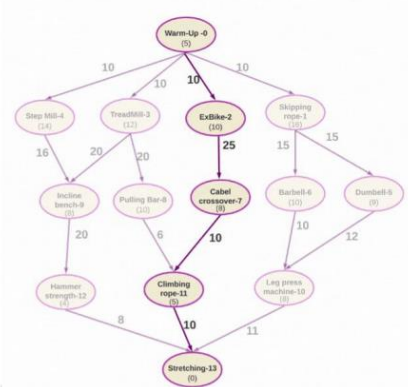
AI in Real-Time Strategy (RTS) Games

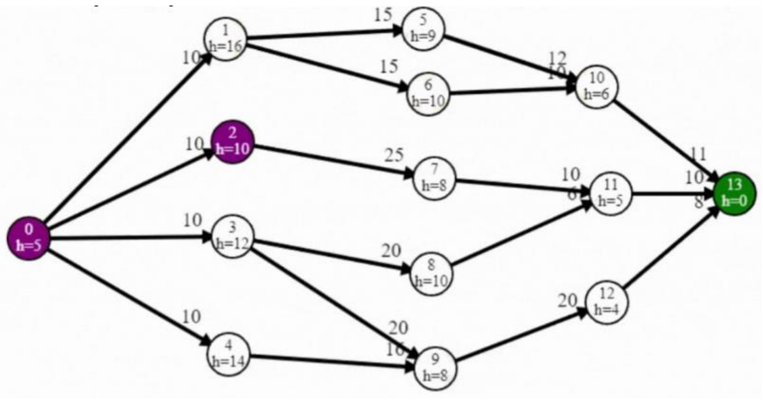
- RTS AI uses techniques like Monte Carlo Tree Search (MCTS) for decision-making under uncertainty.
- Planning and Constraint Satisfaction help AI coordinate resource management and unit actions.
- Machine Learning and reinforcement learning are used to adapt to player strategies.
- Multi-Agent Systems manage multiple units, often using fuzzy logic for decision-making under uncertainty.

Problem solving techniques

1. Heuristics

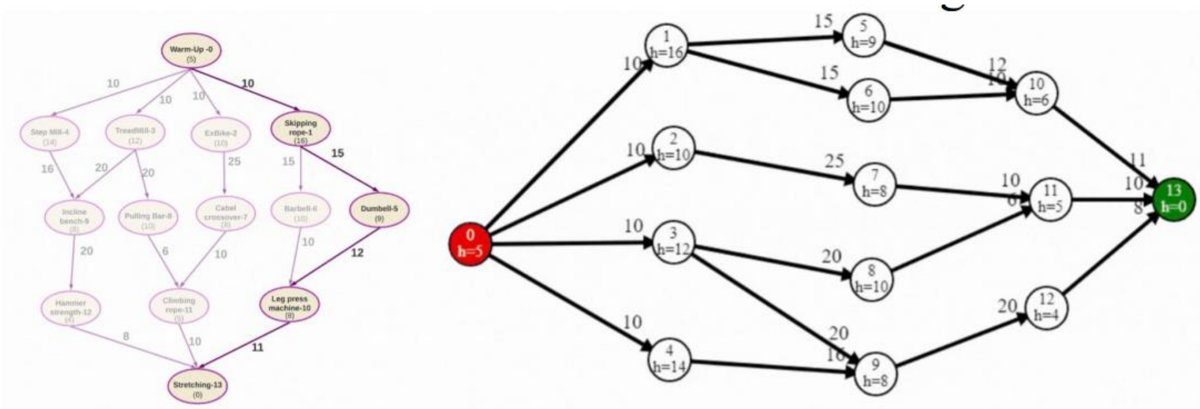
- The heuristic method helps comprehend a problem and devises a solution based purely on experiments and trial and error methods.
- However, these heuristics do not often provide the best optimal solution to a specific problem.
- Instead, these undoubtedly offer efficient solutions to attain immediate goals.
- Therefore, the developers utilize these when classic methods do not provide an efficient solution for the problem.
- Since heuristics only provide time-efficient solutions and compromise accuracy, these are combined with optimization algorithms to improve efficiency.





2. Uninformed/Blind Search

- It is a search algorithm that explores a problem space without any specific knowledge or information about the problem other than the initial state and the possible actions to take.
- It lacks domain-specific heuristics or prior knowledge about the problem.
- Uninformed search applies a way in which search tree is searched without any information, so it is also called blind search.
- It examines each node of the tree until it achieves the goal node.



18. What is Propositional Logic? Explain the logical connectives with examples.

Answer

- Propositional logic (PL) is the simplest form of logic where all the statements are made by propositions. A proposition is a declarative statement which is either true or false. It is a technique of knowledge representation in logical and mathematical form.
- Example:
    a) Normal human body temperature is 98.6F.
    b) The Sun rises from West (False proposition)
    c) 3+3= 7(False proposition)
    d) 5 is a prime number.

Following are some basic facts about propositional logic:

- Propositional logic is also called Boolean logic as it works on 0 and 1.
- In propositional logic, we use symbolic variables to represent the logic, and we can use any symbol for a representing a proposition, such A, B, C, P, Q, R, etc.
- Propositions can be either true or false, but it cannot be both.
- Propositional logic consists of an object, relations or function, and logical connectives.
- These connectives are also called logical operators.

- The propositions and connectives are the basic elements of the propositional logic.
- Connectives can be said as a logical operator which connects two sentences.
- A proposition formula which is always true is called tautology, and it is also called a valid sentence.
- A proposition formula which is always false is called Contradiction.
- A proposition formula which has both true and false values is called contingency.
- Statements which are questions, commands, or opinions are not propositions such as "Where is Rohini", "How are you", "What is your name", are not propositions.

Logical Connectives:

- Logical connectives are used to connect two simpler propositions or representing a sentence logically. We can create compound propositions with the help of logical connectives. There are mainly five connectives, which are given as follows:

1. Negation: A sentence such as $\neg$ P is called negation of P. A literal can be either Positive literal or negative literal.
2. Conjunction: A sentence which has $\wedge$ connective such as, P $\wedge$ Q is called a conjunction. Example: Rohan is intelligent and hardworking. It can be written as,
   P = Rohan is intelligent,
   Q = Rohan is hardworking. $\rightarrow$ P$\wedge$ Q.
3. Disjunction: A sentence which has $\vee$ connective, such as P $\vee$ Q. is called disjunction, where P and Q are the propositions.
   Example: "Ritika is a doctor or Engineer",
   Here P= Ritika is Doctor.
   Q= Ritika is Engineer, so we can write it as P $\vee$ Q.
4. Implication: A sentence such as P $\rightarrow$ Q, is called an implication. Implications are also known as if-then rules. It can be represented as
   If it is raining, then the street is wet.
   Let P= It is raining, and Q= Street is wet, so it is represented as P $\rightarrow$ Q
5. Biconditional: A sentence such as P$\Leftrightarrow$ Q is a Biconditional sentence, example:
   If I am breathing, then I am alive
   P= I am breathing, Q= I am alive, it can be represented as P $\Leftrightarrow$ Q.

19. Explain parsing in NLP. Explain its techniques and applications.

Answer

- Parsing is a crucial technique in Natural Language Processing (NLP) that involves analyzing a string of symbols, either in natural language or computer languages, according to the rules of a formal grammar.
- Parsing helps in understanding the syntactic structure of a sentence and is essential for various NLP tasks such as machine translation, question answering, and text-to-speech systems.

Parsing Techniques in NLP

1. Top-Down Parsing

Top-down parsing is a method of parsing where the process starts from the root of the parse tree (usually the start symbol of a grammar) and tries to break down the sentence into smaller components (like words and phrases), following the grammar rules. The parser predicts what structure the sentence might have and tries to match it with the actual input.

Example

S → NP VP (A sentence is made of a noun phrase and a verb phrase)
NP → Det N (A noun phrase is a determiner followed by a noun) V
P → V NP (A verb phrase is a verb followed by a noun phrase)
Det → 'the‘
N → 'cat' | 'mouse‘
V → 'chases‘

And the sentence to parse: "the cat chases the mouse".

Step-by-Step Top-Down Parsing

1. Start with the start symbol (S):
   The sentence structure is predicted as S → NP VP.
2. Expand NP (Noun Phrase):
   - NP → Det N.
   - Now, predict that the first word in the sentence will be a Det (determiner) followed by a N (noun).
3. Match with input:
   - The input starts with "the", which matches the rule Det → 'the'.
   - Now, look for a noun. The next word in the input is "cat", which matches N → 'cat'.
4. Expand VP (Verb Phrase):
   - VP → V NP.
   - The verb phrase is predicted as a verb followed by a noun phrase.
5. Match with input:
   - The next word in the input is "chases", which matches V → 'chases'.
   - Now, look for another noun phrase (NP).

6. Expand NP:
   - NP → Det N again.
   - The next word is "the", matching Det → 'the', and the last word is "mouse", matching N → 'mouse'.
7. Complete the parse:
   The whole input "the cat chases the mouse" is successfully parsed according to the grammar.

2. Bottom-Up Parsing
   Bottom-Up Parsing is a parsing technique in which the parser starts from the input (the individual words or tokens of a sentence) and tries to build the parse tree by working its way upwards, combining smaller parts (like words or phrases) into larger structures until it reaches the start symbol of the grammar.

How Bottom-up Parsing Works:

- The parser begins with the input words and repeatedly tries to find grammar rules that can combine them into bigger phrases.
- Reduces small parts (like words) into larger parts (like noun phrases or verb phrases) based on grammar rules.
- Continues this process until the whole sentence is reduced to the start symbol of the grammar (e.g., a "sentence").

Example:
Consider the sentence: "The dog barks.“
Grammar Rules:

- S → NP VP (A sentence is made of a noun phrase followed by a verb phrase)
- NP → Det N (A noun phrase is made of a determiner and a noun)
- VP → V (A verb phrase is just a verb)
- Det → 'The'
- N → 'dog'
- V → 'barks'

1. Input: "The dog barks.“
2. Look at the words: Identify parts of speech (Det = "The", N = "dog", V = "barks").
   Det: "The“
   N: "dog“
   V: "barks"
3. Apply the rules bottom-up:
   - First, combine Det + N → NP:
     "The" + "dog" → NP ("The dog")
   - Now combine V → VP:
     "barks" → VP
4. Final reduction:
   Combine NP + VP → S (Sentence):

"The dog" + "barks" → S (The dog barks)

The parser successfully constructs a valid parse tree for the sentence, starting from the individual words and working its way up to form the full sentence.

20. Construct a truth table for the following compound proposition:

Answer

a. (p ∨ q) ∧ ¬ p

Solution:

- p, q : Variables
- (p ∨ q) : p OR q
- ¬p : NOT p
- (p∨q)∧¬p : given compound statement

Truth Table

p	q	p ∨ q	¬p	(p∨q)∧¬p
T	T	T	F	F
T	F	T	F	F
F	T	T	T	T
F	F	F	T	F

- p∨q: This disjunction is true if either p or q is true, or both.
- ¬p: This negation flips the truth value of p
- (p∨q)∧¬p: The final conjunction is true only when both p∨q is true, and ¬p is true.

b) ¬ (p ∧ q) ∨ (r ∧ ¬ q)

- Calculate p∧q (AND operation).
- Calculate ¬(p∧q) (NOT operation on p∧q).
- Calculate ¬q (NOT operation on q).
- Calculate r∧¬q (AND operation).
- Calculate ¬(p∧q)∨(r∧¬q) (OR operation for the final result).

Truth Table:

p	q	r	$p \wedge q$	$\neg(p \wedge q)$	$\neg q$	$r \wedge \neg q$	$\neg(p \wedge q) \vee (r \wedge \neg q)$
T	T	T	T	F	F	F	F
T	T	F	T	F	F	F	F
T	F	T	F	T	T	T	T
T	F	F	F	T	T	F	T
F	T	T	F	T	F	F	T
F	T	F	F	T	F	F	T
F	F	T	F	T	T	T	T
F	F	F	F	T	T	F	T

ANSWER ANY TWO QUESTIONS **(2 X 15= 30)**

21. Explain the concept of knowledge representation in artificial intelligence. Describe various techniques and issues related to knowledge representation.

Answer

- Humans are best at understanding, reasoning, and interpreting knowledge. Human knows things, which is knowledge and as per their knowledge they perform various actions in the real world. But how machines do all these things comes under knowledge representation and reasoning.
- Knowledge representation and reasoning (KR, KRR) is the part of Artificial intelligence which concerned with AI agents thinking and how thinking contributes to intelligent behavior of agents.

Techniques of knowledge representation

There are mainly four ways of knowledge representation which are given as follows:

1. Logical Representation
2. Semantic Networks
3. Production Rules
4. Frames Representation

1. Logical Representation

Logical representation is a language with some concrete rules which deals with propositions and has no ambiguity in representation. Logical representation means drawing a conclusion based on various conditions. This representation lays down some important communication rules. It consists of precisely defined syntax and semantics which supports the sound inference. Each sentence can be translated into logics using syntax and semantics.

Logical representation can be categorized into mainly two logics:

1. Propositional Logics
2. Predicate logics

For example:

- P: "The sky is blue"
- Q: "It is raining" We can represent the statement "If the sky is blue, then it's not raining" as: $P \rightarrow \neg Q$ (If P, then $\neg Q$)

2. Semantic Networks

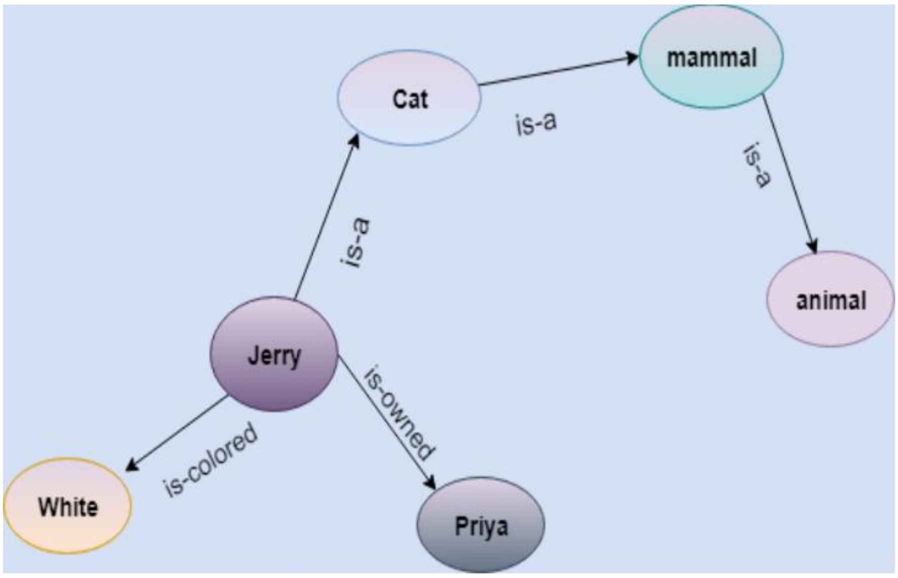
- Semantic networks are alternative of predicate logic for knowledge representation.
- Predicate knowledge, often referred to as predicate logic or first-order logic, a powerful tool for knowledge representation that allows for the

representation of complex relationships, properties, and quantified statements.

- This representation consist of mainly two types of relations:
  1. IS-A relation (Inheritance)
  2. Kind-of-relation

Example: Following are some statements which we need to represent in the form of nodes and arcs. Statements:

1. Jerry is a cat.
2. Jerry is a mammal
3. Jerry is owned by Priya.
4. Jerry is brown colored.
5. All Mammals are animal.



3. Frame Representation

- A frame is a data structure used to represent stereotyped situations, and each frame consists of slots that store specific pieces of information or data. It consists of a collection of slots and slot values. These slots may be of any type and sizes. Slots have names and values which are called facets.
- Facets: The various aspects or characteristics of a slot within a frame is known as Facets. Facets are features of frames which enable us to put constraints on the frames.
- Example: IF-NEEDED facts are called when data of any particular slot is needed. A frame may consist of any number of slots, and a slot may include any number of facets and facets may have any number of values. A frame is also known as slot-filter knowledge representation in artificial intelligence.

Example: Let's take an example of a frame for a book

Slots	Filters
Title	Artificial Intelligence
Genre	Computer Science
Author	Peter Norvig
Edition	Third Edition
Year	1996
Page	1152

4. Production Rules Production rules system consist of (condition, action) pairs which mean, "If condition then action". It has mainly three parts:

- The set of production rules
- Working Memory
- The recognize-act-cycle

In production rules agent checks for the condition and if the condition exists then production rule fires and corresponding action is carried out. The condition part of the rule determines which rule may be applied to a problem. And the action part carries out the associated problem-solving steps. This complete process is called a recognize-act cycle.

Example:

- IF (at bus stop AND bus arrives) THEN action (get into the bus)
- IF (on the bus AND paid AND empty seat) THEN action (sit down).
- IF (on bus AND unpaid) THEN action (pay charges).
- IF (bus arrives at destination) THEN action (get down from the bus).

Knowledge representation issues

Knowledge representation in AI involves how information is structured and stored so that AI systems can utilize it effectively. Some common issues and challenges in knowledge representation include:

1. Expressiveness: Ensuring that the representation language or framework can express all necessary concepts and relationships in the domain accurately. Some domains may require complex relationships or uncertain information, which can be challenging to represent.
2. Efficiency: Representations should be efficient in terms of storage, retrieval, and inference. Large-scale knowledge bases can become computationally expensive to manage and query.

3. Scalability: As knowledge bases grow, ensuring that the representation scales effectively without sacrificing performance is crucial. This involves managing both the size and complexity of the representation.
4. Uncertainty and Vagueness: Many real-world domains involve uncertain or vague information. Representing and reasoning with uncertainty (probabilities, fuzzy logic) is a significant challenge in knowledge representation.
5. Integration of Multiple Sources: AI systems often need to integrate knowledge from various sources, which may use different representation languages or frameworks. Ensuring interoperability and consistency across these sources can be difficult.

22. Describe syntactic analysis in NLP and outline the role of parsing. Explain different parsing techniques and assess their importance in NLP applications.

Answer

- Syntactic analysis or parsing or syntax analysis is the third phase of NLP.
- The purpose of this phase is to draw exact meaning, or you can say dictionary meaning from the text.
- Syntax analysis checks the text for meaningfulness comparing to the rules of formal grammar. For example, the sentence like "hot ice-cream" would be rejected by semantic analyzer.
- In this sense, syntactic analysis or parsing may be defined as the process of analyzing the strings in natural language conforming to the rules of formal grammar.
- The origin of the word 'parsing' is from Latin word 'pars' which means 'part'.
- Parsing is the process of analyzing a sequence of symbols (such as words, tokens, or code) to determine its grammatical structure according to a set of rules or a formal grammar. It is commonly used in programming language processing, natural language processing, and data structure manipulations.
- It is used to implement the task of parsing. It may be defined as the software component designed for taking input data (text) and giving structural representation of the input after checking for correct syntax as per formal grammar.

Parsing Techniques in NLP

1. Top-Down Parsing

Top-down parsing is a method of parsing where the process starts from the root of the parse tree (usually the start symbol of a grammar) and tries to break down the sentence into smaller components (like words and phrases), following the grammar rules. The parser predicts what structure the sentence might have and tries to match it with the actual input.

Example

S → NP VP (A sentence is made of a noun phrase and a verb phrase)
NP → Det N (A noun phrase is a determiner followed by a noun) V
P → V NP (A verb phrase is a verb followed by a noun phrase)
Det → 'the‘
N → 'cat' | 'mouse‘
V → 'chases‘

And the sentence to parse: "the cat chases the mouse".

Step-by-Step Top-Down Parsing

1. Start with the start symbol (S):

   The sentence structure is predicted as S → NP VP.

2. Expand NP (Noun Phrase):
   - NP → Det N.
   - Now, predict that the first word in the sentence will be a Det (determiner) followed by a N (noun).
3. Match with input:
   - The input starts with "the", which matches the rule Det → 'the'.
   - Now, look for a noun. The next word in the input is "cat", which matches N → 'cat'.
4. Expand VP (Verb Phrase):
   - VP → V NP.
   - The verb phrase is predicted as a verb followed by a noun phrase.
5. Match with input:
   - The next word in the input is "chases", which matches V → 'chases'.
   - Now, look for another noun phrase (NP).
6. Expand NP:
   - NP → Det N again.
   - The next word is "the", matching Det → 'the', and the last word is "mouse", matching N → 'mouse'.
7. Complete the parse:

   The whole input "the cat chases the mouse" is successfully parsed according to the grammar.

2. Bottom-Up Parsing

   Bottom-Up Parsing is a parsing technique in which the parser starts from the input (the individual words or tokens of a sentence) and tries to build the parse tree by working its way upwards, combining smaller parts (like words or phrases) into larger structures until it reaches the start symbol of the grammar.

How Bottom-up Parsing Works:

- The parser begins with the input words and repeatedly tries to find grammar rules that can combine them into bigger phrases.
- Reduces small parts (like words) into larger parts (like noun phrases or verb phrases) based on grammar rules.
- Continues this process until the whole sentence is reduced to the start symbol of the grammar (e.g., a "sentence").

Example:
Consider the sentence: "The dog barks.“
Grammar Rules:

- S → NP VP (A sentence is made of a noun phrase followed by a verb phrase)
- NP → Det N (A noun phrase is made of a determiner and a noun)
- VP → V (A verb phrase is just a verb)
- Det → 'The'
- N → 'dog'
- V → 'barks'

1. Input: "The dog barks.“
2. Look at the words: Identify parts of speech (Det = "The", N = "dog", V = "barks").
    Det: "The“
    N: "dog“
    V: "barks"
3. Apply the rules bottom-up:
    - First, combine Det + N → NP:
      "The" + "dog" → NP ("The dog")
    - Now combine V → VP:
      "barks" → VP
4. Final reduction:
    Combine NP + VP → S (Sentence):
        "The dog" + "barks" → S (The dog barks)

The parser successfully constructs a valid parse tree for the sentence, starting from the individual words and working its way up to form the full sentence.

23. Explain the concept of expert systems in artificial intelligence, highlighting their architecture, components, and applications.

Answer

An expert system is a computer program designed to mimic the decision-making ability of a human expert in a specific field.

Expert systems are made up of two parts:

- A knowledge base that stores facts and rules,
- And an inference engine that uses those rules to draw new conclusions
- It uses knowledge and inference rules to solve complex problems, often requiring specialized expertise.

- Expert systems are a key application of artificial intelligence (AI), and they are used in domains such as medical diagnosis, engineering, finance, and more.

Applications of Expert Systems

- Medical Diagnosis: Systems like MYCIN assist doctors in diagnosing infections by suggesting treatments based on symptoms.
- Financial Services: They are used for credit risk evaluation, investment analysis, and fraud detection.
- Manufacturing: They help optimize processes, troubleshoot machinery, and ensure quality control.
- Customer Support: Expert systems power automated customer service systems, providing solutions based on common issues and queries.

Components of an Expert System

1. Knowledge Base:
   - Core component that stores all the domain-specific knowledge needed for the expert system to function. It consists of: Facts, Rules.
2. Inference Engine
   - This component is the "brain" of the expert system.
   - It applies logical reasoning to the knowledge base to draw conclusions or make decisions.
3. User Interface
   - This is the component through which users interact with the expert system.
   - It can be a graphical user interface (GUI), CUI (Character User Interface), or web-based interface.
   - That allows users to input data and receive outputs.
   - Example for CUI, MS-DOS allows the user to navigate, open, and otherwise manipulate files on their computer from a command line instead of a GUI like Windows.
4. Explanation Facility
   - This component provides users with explanations of how the expert system arrived at its conclusions or recommendations.
   - It helps users understand the reasoning process and increases trust in the system.
5. Knowledge Acquisition Module
   - This component is responsible for gathering, updating, and maintaining the knowledge base.
   - It can involve interactions with domain experts to capture their knowledge and ensure the system stays current with new information

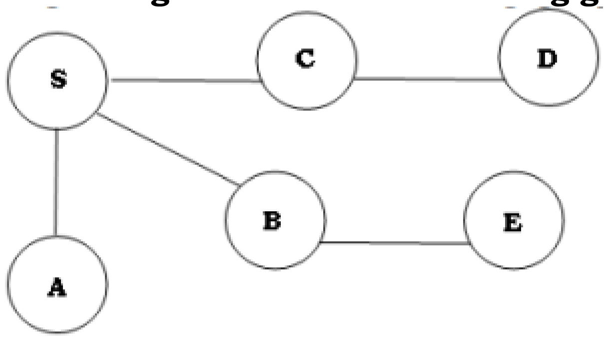
Architecture of Expert Systems

Expert systems simulate human decision-making using these components:

1. Knowledge Base
    - Stores domain-specific facts and rules.
    - Includes factual knowledge (static information) and heuristic knowledge (expert insights).
2. Inference Engine
    - Applies rules to solve problems using forward chaining (data-driven) or backward chaining (goal-driven).
3. User Interface
    - Enables interaction between users and the system for queries and solutions.
4. Knowledge Acquisition
    - Adds new knowledge via expert input or machine learning.
5. Explanation Subsystem
    - Provides reasoning behind decisions.

Example: MYCIN
MYCIN, a medical expert system, diagnosed infections by applying backward chaining to its rule-based knowledge.
This architecture supports applications in healthcare, finance, and more.

24. Write Python code, apply and explain the steps of Breadth First Search algorithm for the following graph



Answer

```
def bfs(graph, start):
    queue = []
    visited = set()
    queue.append(start)
    visited.add(start)
    while queue:
        node = queue.pop(0)
        print(node, end=' ')
        for neighbor in graph[node]:
            if neighbor not in visited:
                queue.append(neighbor)
                visited.add(neighbor)
```

```
graph = {
    'S': ['A', 'B', 'C'],
    'A': [],
    'B': ['E'],
    'C': ['D'],
    'D': [],
    'E': []
}
start_node = 'S'
print("BFS traversal starting from node 'S':", end=' ')
bfs(graph, start_node)
```

Output: BFS traversal starting from node 'S': S A B C E D

BFS Traversal:

1. Start at node S:
    - i. Add S to the queue.
    - ii. Mark S as visited.
2. Visit S's neighbors:
    - i. Dequeue S from the queue.
    - ii. Visit A, B, and C (the neighbors of S).
    - iii. Mark A, B, and C as visited.
    - iv. Add A, B, and C to the queue.
3. Visit A's neighbors:
    - i. Dequeue A from the queue.
    - ii. A has no unvisited neighbors (A's neighbors list is empty).
4. Visit B's neighbors:
    - i. Dequeue B from the queue.
    - ii. Visit E (the neighbor of B).
    - iii. Mark E as visited.
    - iv. Add E to the queue.
5. Visit C's neighbors:
    - i. Dequeue C from the queue.
    - ii. Visit D (the neighbor of C).
    - iii. Mark D as visited.
    - iv. Add D to the queue.
6. Visit E's neighbors:
    - i. Dequeue E from the queue.
    - ii. E has no unvisited neighbors (E's neighbors list is empty).
7. Visit D's neighbors:
    - i. Dequeue D from the queue.
    - ii. D has no unvisited neighbors (D's neighbors list is empty).

At this point, the queue is empty, and the BFS traversal is complete. The order of nodes visited during the traversal is: S, A, B, C, E, D.