AI Unit - 3

Artificial Intelligence UNIT-3 (Comprehensive & Formatted)

Part 1: Learning in Artificial Intelligence

1. What is Learning?

- **Core Definition:** Learning denotes changes in the system that enable it to do the same task more efficiently next time.
- **Alternative Definitions:**
  - Learning is constructing or modifying representations of what is being experienced.
  - Learning is making useful changes in our minds (or the system's knowledge/parameters).
- **Goals/Benefits:**
  - Improves understanding and efficiency.
  - Enables discovery of new things or structures which were previously unknown (e.g., data mining, scientific discovery).
  - Allows filling in skeletal or incomplete observations or specifications about a domain (this expands the domain of expertise and lessens the brittleness of the system).
  - Facilitates building software agents that can adapt to their users or to other software agents.
  - Reproduces an important aspect of intelligent behaviour.

2. What Characterizes a Learning System?

- **Iterative Process:** Machine learning systems often perform the following iteratively:
  1. Produce a result.
  2. Evaluate it against an expected result (if available).
  3. Tweak the system based on the evaluation.
- **Discovery:** Machine learning systems can also discover patterns without prior expected results (unsupervised learning).
- **Transparency Types:**
  - **Open Box:** Changes made to the system (e.g., in the knowledge base) are clearly visible and clearly interpretable by human users.
  - **Black Box:** Changes done to the system are not readily visible or understandable.

3. What is the Architecture of a Learning Agent/System?

- **Main Components:** Machine learning systems typically have four main components:
  - **Knowledge Base (KB):**

        - Stores what is being learnt.
        - Contains the representation of the domain.
        - Includes the description and representation of the problem space.
    - **Performer:** Does something with the knowledge base to produce results.
    - **Critic:** Evaluates the results produced against expected results or performance standards.
    - **Learner:** Takes output (feedback) from the critic and modifies something in the KB or the performer.
- **Optional Component:**
    - **Problem Generator:** May be needed to generate test cases to evaluate performance against.
- **(See Diagram on Slide 6 for Visual Representation of Learning Agent Architecture)**

4. Can you provide Examples of Learning Systems?

Problem	Representation	Performer (interacts with human)	Critic (human player)	Learner (elicits new questions to modify KB)
Animal guessing game	Binary decision tree	Walk the tree and ask associated questions	Human feedback	Elicit a question from the user and add it to the binary tree
Playing chess	The board layout, game rules, moves	Chain through the rules to identify move, use conflict resolution, output	Who won (credit assignment problem)	Increase the weight for some rules and decrease for others
Categorizing documents	Vector of word frequencies, corpus of documents	Apply appropriate functions to identify which category the file belongs to	A set of human-categorized documents	Modify the weights on the function and improve categorization
Fixing computers	Frequency matrix of causes and symptoms	Use known symptoms to identify potential causes	Human input about symptoms and cause	Update the frequency matrix with actual symptoms/outcomes
Identifying digits (OCR)	Probability of digits, matrix of pixels, features (%)	Input the features for a digit, output probability 0-9	Human categorized training set	Modify the weights on the network of associations

5. What are the Different Learning Paradigms?

Paradigm	Description
Rote learning	Knowledge engineering: direct entry of rules and facts; Memorization.
Learning by taking advice	Human/system interaction producing explicit mapping; Advice operationalization.
Learning in problem solving	Parameter adjustments, learning macro-operators, chunking based on experience.
Learning from examples: induction	Using specific examples to reach general conclusions.
Explanation-based learning (EBL)	Learn from a single example and later generalize. More analytical, knowledge-intensive.
Learning through discovery	Unsupervised, specific goal not given; Finding patterns/structures automatically.
Learning through analogy	Determining correspondence between two different representations: case-based reasoning.
Formal learning theory	Formal mathematical model of learning (e.g., PAC learning).
Neural net learning & genetic learning	Evolutionary search techniques, biologically inspired network learning ("survival of fittest").

6. How does Rote Learning work?

- **Definition:** The basic learning activity, also called memorization.
- **Mechanism:** Knowledge, without any modification, is simply copied into the knowledge base. Involves direct entry of rules and facts. The knowledge base *is* captured knowledge.
- **Application:** Traditional approach to develop ontologies; Data caching to improve performance.
- **Benefit:** As computed values are stored, this technique can save a significant amount of time.
- **Use in Complex Systems:** Can be used provided sophisticated techniques are employed to use stored values faster and generalization keeps stored information manageable.
- **Example:** Checkers-playing program storing board positions evaluated during look-ahead search.
- **Key Capabilities Required:**
  - **Organized storage of information:** Need sophisticated techniques for data retrieval, making it faster than recomputing.
  - **Generalization:** Needed to keep the number of distinct objects stored down to manageable levels due to potentially large state spaces.

7. How does Learning by Taking Advice work?

- **Simplicity:** Considered the easiest and simplest way of learning.
- **Process:**
  1. A programmer (or expert) writes instructions/advice for the system.
  2. Once learned (programmed/integrated), the system can do new things based on the advice.

3. Sources can be humans (experts), internet, etc.

- **Inference Requirement:** Requires more inference than rote learning.
- **Operationalization:**
  - Stored knowledge in the KB gets transformed into an *operational form*.
  - The program operationalizes advice by turning it into usable expressions (concepts, actions).
  - This ability to operationalize knowledge is critical (also key in EBL).
- **Consideration:** Reliability of the knowledge source is always taken into consideration.

8. How does Learning Occur in Problem Solving?

- **Context:** Occurs when the program learns by generalizing from its *own experiences*, rather than external advice.
- **Types Discussed:**
  - Learning by parameter adjustment
  - Learning with macro-operators
  - Learning by chunking
  - (Addressing) The Utility Problem
  - **a. How does Learning by Parameter Adjustment work?**
    - **Scenario:** Used when a learning system relies on an evaluation procedure that combines information from several sources (features) into a single summary statistic (score).
    - **Example:** Combining factors like demand and production capacity into a score for increasing production; combining piece advantage and mobility into a score in game programs.
    - **Challenge:** Difficult to know *a priori* how much weight (coefficient) should be attached to each factor.
    - **Representation:** Often uses a polynomial form: Score = $\Sigma\ c_i * t_i$ (where t=feature value, c=weight/coefficient). As learning progresses, 'c' values change.
    - **Process:**
      1. Start with some estimate of the correct weight settings.
      2. Modify the weights based on accumulated experiences.
      3. Features that appear to be good predictors of overall success will have their weights increased, while those that do not will have their weights decreased.
    - **Key Questions/Problems:**
      - *When* should a coefficient be increased/decreased? (Increase for accurate predictors, decrease for poor ones).
      - *The Credit Assignment Problem:* Appropriately assigning responsibility to each step/factor that led to a single outcome.
      - *By how much* should the value be changed?

- **Method:** A variety of hill-climbing search.
- **Usefulness:** Useful when little additional knowledge is available or when combined with more knowledge-intensive methods.

- **b. How does Learning with Macro-Operators work?**
  - **Definition:** Sequences of actions (operators) that can be treated as a whole are called macro-operators (MACROPs).
  - **Process:**
    1. Once a problem is solved, the learning component takes the computed plan (sequence of actions).
    2. Stores it as a single macro-operator.
    3. Preconditions = initial conditions of the problem solved.
    4. Postconditions = goal achieved by the plan.
  - **Benefit:** Allows the problem solver to efficiently use knowledge gained from previous experiences. Critical for problems with non-serializable subgoals (where working on one subgoal interferes with another). One macro can produce a small global change even if individual operators have undesirable local effects. Allows learning domain-specific knowledge.
  - **Generalization:** Generalization (replacing constants with variables) allows macro-operators to solve different but structurally similar problems.
  - **Example:** STRIPS planning algorithm employed macro-operators in its learning phase, building MACROP with preconditions, post-conditions, and the action sequence.

- **c. How does Learning by Chunking work?**
  - **Similarity:** Similar to learning with macro-operators.
  - **Context:** Generally used by problem solvers based on production systems.
  - **Production System Basics:** Consists of IF-THEN rules, a knowledge base, control strategy, and a rule applier. Rules fire when their IF part matches the current situation, executing the THEN part.
  - **Chunking Process:**
    1. Problem solvers apply rules to solve problems.
    2. When a sequence of rule firings proves useful (e.g., solves a sub-problem or resolves an impasse), the results/sequence are stored as a single new rule (a "chunk").
  - **Benefit:** Chunks capture general search control knowledge. Reduces problem-solving steps by replacing a sequence of rule firings with a single chunk firing. Can be used later for the same or similar problems.
  - **Relationship to Macros:** Several chunks might encode a single macro-operator, or one chunk might participate in multiple macro sequences.
  - **Example: SOAR Architecture:**
    - A general cognitive architecture. Acquires knowledge via chunking.

        - Learns reflexively when *impasses* (lack of knowledge to proceed) are resolved.
        - Resolving an impasse involves exploring a new problem space; the steps taken are chunked.
        - Chunks are generalized productions stored in long-term memory, decreasing future search effort for similar impasses.
    - **d. What is the Utility Problem in Learning?**
        - **Definition:** Occurs when knowledge learned in an attempt to *improve* a system's performance *degrades* it instead.
        - **Context:** Familiar in speedup learning systems (which learn control rules to guide problem-solving). Often, systems slow down if allowed to learn unrestrainedly.
        - **Paradox:** Each individual control rule might be guaranteed to have positive utility (improve performance in isolation), but in concert, they have negative utility (degrade overall performance).
        - **Cause Example:** The serial nature of hardware. More control rules acquired means longer time taken to test/match them on each cycle.
        - **Solutions:**
            - **Hardware:** Design parallel memory systems ("active memories") to eliminate increased match cost by moving matching away from the central processor.
            - **Utility Measurement (e.g., PRODIGY program):**
                1. Maintain a utility measure for each control rule (considering average savings, application frequency, match cost).
                2. Discard proposed rules with negative utility.
                3. Monitor the utility of learned rules during subsequent problem solving.
                4. Discard rules if their utility falls below a threshold.
                    - Empirical experiments show the effectiveness of keeping only high-utility rules.
            - **Applicability:** Utility considerations apply to a wide range of learning problems.

9. How does Learning by Analogy work?

- **Definition:** Acquiring new knowledge about an input entity by transferring it from a known similar entity.
- **Central Intuition:** If two entities are similar in some respects, they could be similar in other respects as well.
- **Examples:**
    - Simple Hydraulics Problem vs. Kirchoff's First Law (inferring similar laws). Pressure Drop ~ Voltage Drop.
    - A variable in a programming language is like a box.
- **Types:**

- **Transformational Analogy:**
  - Look for a similar *solution*.
  - Copy it to the new situation, making suitable substitutions where appropriate.
  - Focuses only on the final solution, not the derivation steps.
  - **Example (Geometry):** Knowing `RO = NY` and `angles AOB = COD`. Seeing the additive rule `RO + ON = ON + NY` implies `RN = OY`. By analogy, apply the same additive logic to angles: `AOB + BOC = BOC + COD` implies `AOC = BOD`. (See Slide 23 diagram).
- **Derivational Analogy:**
  - Finds problems sharing significant aspects based on a similarity metric/threshold.
  - Retrieves the *derivation* (history/steps) of the previous solution.
  - Perturbs the *old derivation* incrementally until it satisfies the new problem's requirements.
  - Considers *how* the problem was solved.
  - **Example (Geometry):** Given a proof for `AB = CD` => `AC = BD`, use the *steps* of that proof (adding BC to both sides) to prove the analogous angle problem `<BAC = <DAE` => `<BAD = <CAE` (by adding <CAD to both sides). (See Slides 24-25 diagrams).

10. What is Explanation-Based Learning (EBL)?

- **Definition:** Learning from a single example by:
  1. **Explaining:** Using existing domain knowledge (a theory) to explain *why* the training example is an instance of the target concept.
  2. **Generalizing:** Turning the explanation into a more general rule or concept definition that captures the essential features demonstrated by the example and explanation.
- **Approach:** Analytical and knowledge-intensive (requires a good domain theory).
- **(Note: Slides 26-31 in the source material appear to be placeholders or diagrams intended to illustrate the EBL process, likely showing steps like constructing an explanation proof tree, generalizing the tree, and forming a new rule.)**

11. How does Learning by Discovery work?

- **Definition:** An entity acquires knowledge without the help of a teacher (unsupervised learning).
- **Types Discussed:**
  - **Theory-Driven Discovery (e.g., AM program - 1976):**
    - **Goal:** Discovers concepts in elementary mathematics and set theory.
    - **Inputs:**
      - Description of some set theory concepts (LISP form, e.g., set union, intersection).
      - Information on how to perform mathematics (e.g., functions).
    - **How it Works:**
      - Uses frame-based representation for concepts (can create new slots/fill values).

        - Employs heuristic search (guided by ~250 heuristics about potentially interesting activities like employing functions, creating concepts, generalization).
        - Uses Hypothesis and Test based search.
        - Agenda control manages the discovery process.
    - **Discoveries:** Integers (via counting set elements), Addition (via counting union of disjoint sets), Multiplication, Prime Numbers (via factorization), Goldbach's Conjecture (even numbers as sum of 2 primes), Maximally Divisible Numbers (k has more factors than any integer < k).
- **Data-Driven Discovery (e.g., BACON program - 1981):**
    - **Context:** Many discoveries arise from observing and making sense of empirical data (e.g., Astrophysics, Quantum Mechanics).
    - **How it Works:**
        1. Starts with a set of variables for a problem (e.g., p, V, n, T for ideal gas law).
        2. Inputs values from experimental data.
        3. Holds some variables constant and attempts to notice trends/relationships in the data among the varying ones.
        4. Makes inferences (mathematical laws).
    - **Discoveries:** Ideal gas law (pV/nT = k), Kepler's 3rd law, Ohm's law, conservation of momentum, Joule's law.
- **Clustering:**
    - **Definition:** A common descriptive task involving grouping data into several new classes or clusters. The process of grouping physical or abstract objects into classes of similar objects.
    - **Cluster:** A collection of data objects similar to one another within the same cluster and dissimilar to objects in other clusters.
    - **Goal:** Construct meaningful partitioning; maximize intra-class similarity and minimize inter-class similarity.
    - **Process:** Given N k-dimensional feature vectors, find a "meaningful" partition into 'c' subsets. Discover labels automatically. 'c' may be given or discovered. More difficult than classification (where groups are pre-defined).
- **AutoClass (Clustering Algorithm Example):**
    - **Approach:** A clustering algorithm based on the Bayesian approach for determining optimal classes in large datasets.
    - **Goal:** Given data X={X1, …, Xn} with unknown classes, search for the best class description (model) that predicts the data.
    - **Mechanism:**
        - Class membership is expressed *probabilistically*. An instance isn't assigned to a unique class but has a probability (weight) of belonging to each possible class.

        - Calculates the likelihood of each instance belonging to each class C.
        - Calculates weights `wij = P(Ci | Xj)` for each instance `Xj` and potential class `Ci`.
        - Uses weighted statistics (based on these weights) relevant to each term of the class likelihood to estimate the class model parameters.
        - The classification step (computing weights and parameters) is computationally intensive.

12. What is Formal Learning Theory?

- **Focus:** Provides a formal mathematical model of learning. Analyzes learnability.
- **Example Theory: Theory of the Learnable (Valiant - leading to PAC Learning):**
    - Classifies problems by how difficult they are to learn.
    - Formally, a device can learn a concept if it can, given positive and negative examples, produce an algorithm (hypothesis) that will classify future examples correctly with probability `1 - ε` (where `ε` is the error tolerance, related to `1/h` in the notes' notation).
- **Complexity Factors:** The complexity/difficulty of learning a function depends on:
    - The error tolerance (`ε` or `h`).
    - The number of features (e.g., binary features `t`).
    - The complexity/size (`f`) of the rules/hypothesis needed for discrimination.
- **Trainability:** If the number of training examples required is *polynomial* in these factors (1/ε, t, f), the concept class is considered efficiently learnable (trainable).
- **Goal:** Aims to quantify the use of knowledge in learning from a mathematical perspective.

13. What are Neural Net Learning and Genetic Learning?

- **(Note: Slides 40-41 provide minimal explicit detail beyond listing them.)**
- **Neural Net Learning:**
    - Biologically inspired by brain structure.
    - Uses networks of interconnected processing units (artificial neurons).
    - Learning occurs primarily by adjusting the strengths (weights) of connections between neurons based on training data and error feedback (e.g., backpropagation).
    - Falls under parameter adjustment but uses specific network architectures (MLPs, CNNs, RNNs, Transformers etc.).
- **Genetic Learning (Genetic Algorithms):**
    - Inspired by biological evolution and "survival of the fittest".
    - Belongs to the class of evolutionary search techniques.
    - Maintains a population of candidate solutions.
    - Uses operators like fitness evaluation, selection (survival of better solutions), crossover (combining parts of solutions), and mutation (random changes) to iteratively evolve better solutions.

Part 2: Game Playing in AI

14. What is the Minimax Algorithm?

- **Type:** A backtracking algorithm used for decision making in game theory and AI.
- **Application:** Used in two-player, zero-sum, perfect information games (e.g., Tic-Tac-Toe, Checkers, Chess).
- **Goal:** To find the optimal move for a player (Maximizer - MAX), assuming the opponent (Minimizer - MIN) also plays optimally.
- **Players:**
  - **Maximizer (MAX):** Tries to get the highest score possible (maximum benefit).
  - **Minimizer (MIN):** Tries to get the lowest score possible (minimum benefit for MAX, maximum benefit for MIN).
- **Evaluation:** Every game board state (node in the game tree) can be assigned an evaluation score. Positive score often favors MAX, negative favors MIN.

15. How does the Minimax Algorithm Work?

1. **Generate Game Tree:** Generate the tree of possible moves and resulting states down to a certain depth or until terminal states (game end) are reached.
2. **Apply Utility Function:** Assign scores (utility values) to the terminal nodes (leaf nodes).
3. **Backtrack and Propagate Values:**
   - Apply Depth-First Search (DFS) to traverse the tree down to the leaves.
   - Recursively compute values for nodes moving upwards from the leaves.
   - **At Terminal Nodes:** Use the utility function value.
   - **At MAX Nodes:** Choose the MAXIMUM value among its children nodes.
   - **At MIN Nodes:** Choose the MINIMUM value among its children nodes.
   - Continue until the root node (current state) is reached. The value at the root is the best achievable score for MAX assuming optimal play. The move leading to this value is chosen.

- **Example Workflow:**
  - **(Step 1 - Initial State):** Start at root (A). Assume MAX plays first. Generate tree to terminal nodes. MAX's initial best is -∞, MIN's is +∞.
  - **(Step 2 - Evaluate Leaf Children for MAX Layer):** Calculate values for nodes (D, E, F, G) assuming they are MAX's choice from terminal leaves below them.
    - Node D: max(-1, 4) = 4
    - Node E: max(2, 6) = 6
    - Node F: max(-3, -5) = -3
    - Node G: max(0, 7) = 7

- **(Step 3 - Evaluate MIN Layer):** Calculate values for MIN nodes (B, C) using the values from Step 2.
  - Node B: min(Value(D), Value(E)) = min(4, 6) = 4
  - Node C: min(Value(F), Value(G)) = min(-3, 7) = -3
- **(Step 4 - Evaluate MAX Root Layer):** Calculate value for the root MAX node (A) using values from Step 3.
  - Node A: max(Value(B), Value(C)) = max(4, -3) = 4
- **(Result):** The optimal value for MAX is 4. MAX should choose the move leading to state B.

16. What are the Properties of Minimax?

- **Completeness:** Complete if the game tree is finite. It will definitely find a solution (if one exists).
- **Optimality:** Optimal if both opponents are playing optimally.
- **Time Complexity:** $O(b^m)$, where 'b' is the branching factor and 'm' is the maximum depth. Exponential.
- **Space Complexity:** O(b*m) for DFS (stores the path).

17. What is the Limitation of the Minimax Algorithm?

- **Slow Performance:** Gets really slow for complex games (Chess, Go) with huge branching factors and depths due to the exponential time complexity. It explores the entire tree up to depth 'm'.

18. What is Alpha-Beta Pruning?

- **Definition:** A modified version of Minimax; an optimization technique.
- **Goal:** Reduce the number of nodes examined by Minimax while returning the same optimal move. It achieves this by *pruning* (eliminating) branches that cannot possibly influence the final decision.
- **Mechanism:** Uses two threshold parameters, Alpha (α) and Beta (β), to track bounds on possible scores.
  - **Alpha (α):** The best (highest-value) choice found *so far* at any point along the path for the **Maximizer**. Initial value: -∞.
  - **Beta (β):** The best (lowest-value) choice found *so far* at any point along the path for the **Minimizer**. Initial value: +∞.
- **Pruning Condition:** Pruning occurs when **α ≥ β**. If this condition is met at a node, the remaining children of that node do not need to be explored.
- **Scope:** Can be applied at any depth and can prune single leaves or entire sub-trees.

19. What are the Key Points and Working of Alpha-Beta Pruning?

- **Key Points:**
  - The MAX player will only update the value of alpha.
  - The MIN player will only update the value of beta.

    - While backtracking, the *node's computed value* (min or max of children) is passed upwards, not the alpha/beta values themselves.
    - Alpha and beta values are passed *down* to child nodes during exploration.
- **Condition for Pruning:** $\alpha \geq \beta$
- **Working Example Steps:**
    1. **Start at Root (A):** $\alpha = -\infty$, $\beta = +\infty$. Pass down to B.
    2. **Node B:** $\alpha = -\infty$, $\beta = +\infty$. Pass down to D.
    3. **Node D (MAX turn from leaves):** Evaluate leaves. max(-∞, 2)=2 -> α=2. max(2, 3)=3 -> α=3. Node D value is 3. Return 3 to B.
    4. **Node B (MIN turn):** Update β. β = min(+∞, 3) = 3. Now B has $\alpha = -\infty$, $\beta = 3$. Pass these down to E.
    5. **Node E (MAX turn):** $\alpha = -\infty$, $\beta = 3$. Evaluate first child (5). α = max(-∞, 5) = 5. **Check Pruning:** Now α (5) ≥ β (3). **Prune!** Do not evaluate the right child of E. Node E value is 5. Return 5 to B.
    6. **Node B (MIN turn finishes):** Update β. β = min(3, 5) = 3. Node B's final value is 3. Return 3 to A.
    7. **Node A (MAX turn):** Update α. α = max(-∞, 3) = 3. Now A has $\alpha = 3$, $\beta = +\infty$. Pass these down to C.
    8. **Node C:** $\alpha = 3$, $\beta = +\infty$. Pass down to F.
    9. **Node F (MAX turn from leaves):** $\alpha = 3$, $\beta = +\infty$. Evaluate left child (0). α = max(3, 0) = 3. Evaluate right child (1). α = max(3, 1) = 3. Node F value is 1. Return 1 to C.
    10. **Node C (MIN turn):** Update β. β = min(+∞, 1) = 1. Now C has $\alpha = 3$, $\beta = 1$. **Check Pruning:** α (3) ≥ β (1). **Prune!** Do not evaluate child G. Node C value is 1. Return 1 to A.
    11. **Node A (MAX turn finishes):** Update α. α = max(3, 1) = 3. Root node A's final value is 3.
    12. **Result:** The optimal value for MAX is 3. The pruning eliminated exploration of one leaf under E and the entire subtree under G.

20. How does Move Ordering Affect Alpha-Beta Pruning?

- **High Dependence:** The effectiveness (amount of pruning) is highly dependent on the order in which nodes (moves) are examined.
- **Worst Ordering:**
    - Occurs when the best moves are examined last at each node.
    - Results in minimal or no pruning; the algorithm works almost exactly like Minimax (but with overhead of checking α/β).
    - Time complexity remains $O(b^m)$.
- **Ideal Ordering:**
    - Occurs when the best move is always examined first at each node.

- Leads to maximum pruning. Allows the search to go roughly twice as deep as Minimax in the same amount of time.
- Time complexity approaches $O(b^{m/2})$.

- **Rules to Find Good Ordering:**
  - Try to explore the best move from the shallowest node first.
  - Order nodes such that potentially best ones are checked first (using heuristics).
  - Use domain knowledge (e.g., Chess: captures first, then threats, then forward moves, backward moves).
  - Use techniques like iterative deepening and storing results from previous searches (transposition tables/bookkeeping) to guide ordering in subsequent searches.

Part 3: Natural Language Processing (NLP)

(Initial Content from Slide 43)

Presenter: Asif Ekbal, Dept. of Computer Science and Engineering, IIT Patna
Topic: Introduction to Natural Language Processing

21. What is Natural Language Processing (NLP)?

- **Definition:** NLP is the branch of computer science (and AI) focused on developing systems that allow computers to communicate with people using everyday, natural language.
- **Related Field:** Computational Linguistics (also concerns how computational methods can aid the understanding of human language).
- **Two Goals:**
  - **Science Goal:** Understand the way language operates.
  - **Engineering Goal:** Build systems that analyse and generate language; reduce the man-machine gap.
- **Two Views:**
  - Classical View (Symbolic, Rule-based)
  - Statistical/Machine Learning View (Data-driven)

22. What are the Core Areas/Levels of NLP (The NLP Trinity/Stages)?

- **(Diagram on Slide 44 illustrates complexity increase)**
- **Morphological Analysis:** Word structure (morphemes).
- **Part-of-Speech (POS) Tagging:** Assigning word categories (noun, verb).
- **Chunking/Shallow Parsing:** Identifying basic phrases (Noun Phrases, Verb Phrases).
- **Parsing/Syntactic Analysis:** Determining full sentence structure (grammar).
- **Semantics:** Extracting meaning (word sense, sentence meaning, semantic roles).

- **Discourse & Coreference:** Analyzing relationships between sentences, resolving pronouns.
- **Algorithms/Models Used:** HMM, MEMM, CRF, RNN (examples shown on diagram).
- **Languages:** Applicable to various languages (English, Hindi, French, Marathi shown).

23. Why is NLP/Language Technology Relevant in the Indian Context?

- **Linguistic Diversity:** India is highly multilingual.
  - 22 official languages (constitution).
  - Census 2001: 122 major languages, 1599 other languages, 2371 scripts.
  - 30 languages spoken by >1 million; 122 spoken by >10,000.
  - Major language families: Indo-European, Dravidian, Sino-Tibetan, Austro-Asiatic.
  - Hindi (~350m), Bangla (~230m), Marathi (~84m) are globally ranked.
- **Digital Divide:** Only ~20% understand English, leaving 80% potentially excluded from digital content/services if not available in local languages.
- **Internet/Social Media Growth:** Phenomenal growth in users and increasing use of Indian languages online necessitates NLP for processing this content.
- **Goal:** Language Technology/NLP is crucial for effective communication, creating an inclusive digital society, and tackling the digital divide by allowing citizens flexibility in their own languages.

24. What are some Indian Government Initiatives in NLP (e.g., TDIL)?

- **TDIL Programme (Technology Development for Indian Languages - MeiTY):**
  - **Objective:** Develop Information Processing Tools & Techniques to facilitate human-machine interaction without language barriers; create and access multilingual knowledge resources; integrate them for innovative user products/services.
- **Major Initiatives:**
  - **Machine Translation (MT):**
    - *Anuvadaksh:* English to Hindi/Marathi/Bangla/Oriya/Tamil/Urdu/Gujrati/Bodo.
    - *Angla-Bharti based:* English to Bangla/Punjabi/Malayalam/Urdu/Hindi/Telugu.
    - *Sampark (Indian Lang to Indian Lang):* 18 pairs (e.g., Hindi<>Bengali, Hindi<>Marathi, Tamil<>Telugu).
  - **Cross-Lingual Information Access (CLIA):** Assamese, Bengali, Hindi, Oriya, Punjabi, Tamil, Telugu, Marathi.
  - **OCR (Robust Document Analysis & Recognition):** 14 languages (Assamese, Bengali, Devanagari, Gujarati, Gurumukhi, Kannada, Malayalam, Manipuri, Marathi, Oriya, Tamil, Telugu, Tibetan, Urdu).
  - **Text-to-Speech (TTS):** In Indian Languages.
  - **Automatic Speech Recognition (ASR):** In Indian Languages.
  - **Domain-Specific MT:** Hindi to English (Judicial Domain).

25. How is NLP used in Governance (Case Study: MyGov.in Portal)?

- **MyGov.in Portal:**
  - Citizen-centric platform for connecting people with Government for participatory governance.
  - Launched July 26, 2014 by PM.
  - Goal: Bring government closer, exchange ideas, contribute to socio-economic transformation.
  - Features: Discussions, Tasks, Talks, Polls, Blogs on various policy issues (Clean Ganga, Girl Child Education, Skill Development etc.).
  - Success: >1.78 Million users, >10,000 posts/week. Feedback analyzed for actionable suggestions.
- **NLP Challenge:** Infeasible to manually mine relevant info from huge volume of user posts. Need automated analysis.
- **Code-Mixing Issue:** Users often mix languages (e.g., Hindi + English) in posts, requiring NLP techniques robust to this phenomenon.
  - *Example:* "Kolkata to Varanasi ka kya distance hai"
- **Need for NLP/ML:** Sophisticated NLP and ML techniques are demanded to:
  - Analyze feedback automatically.
  - Understand public opinion (Sentiment Analysis).
  - Handle code-mixed data.

26. Why Analyse Public Opinion using NLP?

- **Importance:** Public opinions play important roles for the betterment of human lives.
- **Opportunity:** Huge volumes/varieties of user-generated content offer new opportunities to understand social behavior.
- **Benefit for Governance:** Understanding deep public feelings can help government anticipate social changes and adapt to population expectations.
- **Relevant Field:** Opinion Mining or Sentiment Analysis.

27. What is the Projected Growth and Evolution of NLP?

- **Growth:** Growing exponentially. Expected market of $16 billion by 2021, with ~16% annual compound growth.
- **Reasons for Growth:**
  - Rise of Chatbots.
  - Need for customer insights.
  - Automation of messaging/support.
  - Content translation.
  - Automation of language/speech tasks.
- **Major Industry Players:** Amazon, Google, Microsoft, Facebook, IBM etc.

- **Evolution:** Evolving from human-computer interaction to human-computer *conversation*.
- **Critical Advancements:** Biometrics, Humanoid Robotics.

28. How can NLP be used specifically in Governance?

- **Goal:** Improve service delivery, decrease citizen-government interaction gap.
- **Uses on Government Websites:**
  - Making e-governance info available in multiple languages (MT).
  - Natural Language Generation (NLG) for reports, summaries.
  - Chatbots for information access and service requests (can support multilingual interaction, even for non-literate users via speech).

29. How can NLP be used in Business and Healthcare?

- **Business:**
  - **Sentiment Analysis:** Analyzing public/customer opinion on products/services.
  - **Email Filters:** Filtering spam, categorizing emails.
  - **Voice Recognition:** Smart voice-driven interfaces/services.
  - **Information Extraction:** Extracting key data from documents.
- **Healthcare:**
  - **Improved EHR Experience:** Analyzing Electronic Health Records, clinical notes. Voice-support, predictive/prescriptive analytics.
  - **Reduced Communication Gap:** Allowing patient interaction (e.g., with portals) in their own language. Easier understanding of health status.
  - **Improved Quality of Care:** Calculating inpatient care measures, monitoring clinical guidelines from reports.
  - **Patient Identification:** Identifying patients needing improved care coordination (e.g., automated cancer detection, identifying root causes of substance disorders).

30. How can NLP be used in Finance and Other Domains?

- **Finance:**
  - **Credit Scoring:** Using ML/NLP to assess creditworthiness from various data sources (e.g., Lenddo Score).
  - **Document Search/Analysis:** Automating documentation processing (e.g., Nuance Document Finance Solution).
  - **Fraud Detection:** Analyzing transactions/communications for fraud patterns.
  - **Stock Market Prediction:** Using sentiment analysis on news/social media.
- **Other Domains:**
  - **National Security:** Sentiment analysis in cross-border languages, hate speech/radicalization detection.

- **Recruitment:** Searching/screening applications/resumes, selecting best candidates.

31. What are the Perspectives and Allied Disciplines of NLP?

- **AI Inter-dependencies:** NLP interacts with Search, Logic, ML, Vision, Knowledge Representation, Planning, Robotics, Expert Systems. (See Diagram Slide 49).
- **Allied Disciplines:**
  - **Philosophy:** Semantics, meaning, logic.
  - **Linguistics:** Syntax, lexicon, semantics, language structure.
  - **Probability & Statistics:** Corpus linguistics, hypothesis testing, evaluation.
  - **Cognitive Science:** Computational models of language processing, acquisition.
  - **Psychology:** Behaviouristic insights, psychological models.
  - **Brain Science:** Language processing areas in the brain.
  - **Physics:** Information Theory, Entropy.
  - **Computer Science & Engineering:** Building NLP systems.

32. What is the Turing Test?

- **(Alan Turing, 1950)** A test of a machine's ability to exhibit intelligent behaviour equivalent to, or indistinguishable from, that of a human.
- **Setup:** A human judge engages in natural language conversations with a human and a machine, both concealed. If the judge cannot reliably tell which is which, the machine is said to have passed the test. Based on language interaction. (See Diagram Slide 34).

33. How do Natural Languages Differ from Computer Languages?

- **Ambiguity:** The primary difference. Natural languages are inherently ambiguous at multiple levels.
- **Formal Languages (Programming):** Designed to be *unambiguous*. Can be defined by a grammar producing a unique parse (generally). Designed for efficient, deterministic parsing (often Deterministic Context-Free Languages - DCFLs, parsable in O(n) time).

34. What are the Stages of NLP Processing and Ambiguity Examples?

- **(NLP architecture involves stages with ambiguity at each)**
- **Phonetics & Phonology (Speech Processing):**
  - *Challenges:* Homophones (bank/bank), Near Homophones (maatraa/maatra), Word Boundary segmentation (aajaayenge -> aa jaayenge / aaj aayenge; I got [ua]plate), Phrase boundary, Disfluencies (um, ah).
- **Morphological Analysis (Word Structure):**
  - *Definition:* Study of internal word structure. Morpheme is smallest meaningful unit (e.g., carry, pre, ed, ly, s).
  - *Task:* Segmenting words into morphemes (carried -> carry + ed).

- *Challenge:* Ambiguity (unlockable -> un + lockable OR unlock + able?). Rich morphology in Indian languages vs. poor in English/Chinese.
- *Rules:* Word formation (plural, gender, tense, aspect, modality).

- **Lexical Analysis (Dictionary Lookup):**
  - *Task:* Access dictionary, get word properties (POS, semantic features like animate).
  - *Challenge: Lexical Disambiguation:*
    - Part-of-Speech Disambiguation (dog noun vs. dog verb).
    - Word Sense Disambiguation (WSD) (dog animal vs. dog person). Requires context.
  - *Challenge: Neologisms/New Senses:* Tech terms (justify margin), new verbs (Xeroxed), new expressions (digital trace, communifaking, discomgooglation, helicopter parenting).
- **Syntactic Analysis (Sentence Structure / Parsing):**
  - *Task:* Detect structure (e.g., using grammar rules S->NP VP). (See Tree Diagram Slide 49).
  - *Challenge: Structural Ambiguity:*
    - Scope: "old men and women"
    - Prepositional Phrase Attachment: "I saw the boy with a telescope" (Requires world knowledge to resolve: "mountain with telescope", "boy with pony-tail"). Ubiquitous (newspaper headlines).
  - *Challenge: Garden Path Sentences:* Grammatically correct but misleading initial structure ("The horse raced past the garden fell.").
- **Semantic Analysis (Meaning Representation):**
  - *Task:* Represent meaning (e.g., Predicate Calculus, Semantic Nets, Frames). Identify semantic roles (Agent, Patient, Instrument). (E.g., "John gave a book to Mary").
  - *Challenge: Semantic Role Labeling (SRL) Ambiguity:* "Visiting aunts can be a nuisance". Language differences ("aapko mujhe mithaai khilaanii padegii").
  - *Challenge: Word Sense Disambiguation (WSD):* Choosing the correct meaning of a word in context ("strong interest" vs. "pay interest").
- **Pragmatics (Contextual Meaning / User Intent):**
  - *Task:* Model user intention, understand meaning beyond literal interpretation. Requires world knowledge.
  - *Challenge:* Very hard. Example: Tourist asking boy about sandals implies wanting them brought down, not just confirmation. Understanding headlines ("WHY INDIA NEEDS A SECOND OCTOBER?").
- **Discourse (Multi-Sentence Analysis):**
  - *Task:* Processing sequences of sentences. Understanding relationships between them.
  - *Challenge: Anaphora / Co-reference Resolution:* Determining what pronouns/phrases refer to ("John put the carrot on the plate and ate *it*.", "Bush started the war... *the president* needed

consent..."). Requires reasoning ("Jack has a kite. He will make *you* take *it* back.").

- *Challenge: Ambiguity resolution* requires complex reasoning and world knowledge across sentences.

35. What are some Specific NLP Tasks?

- **Word Segmentation:** Breaking character strings into words (crucial for languages like Chinese, relevant for URLs/hashtags).
- **Part-of-Speech (POS) Tagging:** Annotating each word with its POS tag. Useful for parsing and WSD.
- **Phrase Chunking:** Finding non-recursive NPs and VPs.
- **Parsing:** Full syntactic analysis, generating a parse tree.
- **Word Sense Disambiguation (WSD):** Determining the correct meaning of ambiguous words in context.
- **Semantic Role Labeling (SRL):** Identifying the semantic role of noun phrases relative to verbs (agent, patient, etc.). Also called case role analysis, thematic analysis, shallow semantic parsing.
- **Textual Entailment:** Determining if one sentence logically entails (implies) another. (See PASCAL Challenge examples, Slide 59).
- **Anaphora Resolution / Co-reference Resolution:** Identifying phrases referring to the same entity.
- **Information Extraction (IE):**
  - **Named Entity Recognition (NER):** Identifying names (people, places, organizations).
  - **Relation Extraction:** Identifying relations between entities.
- **Question Answering (QA):** Answering natural language questions using a text corpus.
- **Text Summarization:** Producing short summaries of longer documents.
- **Sentiment Analysis:** Extracting subjective information, polarity (positive/negative), opinions.
- **Machine Translation (MT):** Translating between languages. Requires resolving ambiguities. (Apocryphal examples: "spirit is willing..." -> "liquor is good..."; "out of sight..." -> "invisible idiot").

36. Why is Ambiguity Resolution Hard and What Knowledge is Needed?

- **Requirement:** Correct interpretation requires combining knowledge from multiple levels:
  - **Syntax:** Grammatical structure (e.g., agent is often subject).
  - **Semantics:** Word meanings (people names, city names, company/brand names).
  - **Pragmatics:** Contextual understanding.
  - **World Knowledge:** Common sense facts (credit cards have interest, hammers aren't animate).

37. How has the Approach to Acquiring Knowledge for NLP Changed?

- **Traditional / Rationalist Approach:**
  - Relied on manual knowledge acquisition. Human specialists specified and formalized rules (lexicons, grammars).

    - **Problems:** Difficult, time-consuming, error-prone. Rules have exceptions ("All grammars leak." - Sapir 1921). Systems were expensive, limited, and brittle (not robust).
- **Modern / Empirical / Statistical / Learning Approach:**
    - Uses machine learning to *automatically acquire* knowledge from data (annotated text corpora).
    - Dominant since the 1990s (started earlier in speech recognition).
    - **Process:** Labeled data -> ML Algorithm -> NLP System (with learned linguistic knowledge). (See Diagram Slide 75).
    - **Benefits:** More robust, adaptable, handles variability better.

38. What are Key Milestones in NLP History?

- **1950s:** Shannon (probabilistic models), Chomsky (formal grammars), first parsers (Harris, Joshi), Bayesian OCR.
- **1960s:** MIT AI Lab work (BASEBALL, ELIZA), semantic nets (Simmons), Brown Corpus (Kucera/Francis), Bayesian authorship (Mosteller/Wallace).
- **1970s:** Deeper understanding systems (SHRDLU, Schank's scripts/plans), Prolog for parsing, early HMMs for speech (Baker, Jelinek).
- **1980s:** More complex grammars (unification, TAG), symbolic discourse/generation, initial statistical POS tagging (Church).
- **1990s: "Statistical Revolution"**: Rise of empirical methods, annotated corpora (Penn Treebank etc.), statistical MT (IBM), robust statistical parsers (Magerman, Collins, Charniak), IE systems (MUC).
- **2000s:** Diverse ML methods (SVMs, MaxEnt, CRFs), more shared tasks/corpora (TREC Q/A, SENSEVAL/SEMEVAL, CONLL), unsupervised/semi-supervised learning focus, shift towards semantics (WSD, SRL).
- **2000s onwards / 2010s - Present:** IE from social networks, IR, CLIA, Statistical/Neural MT, Biomedical text mining, Discourse processing, **Deep Learning dominates** (Word2vec, GloVe, CNNs, RNNs, Transformers like ELMo/BERT), focus on end-to-end learning, large pre-trained models, Explainable AI (XAI). **Turing Award 2018** to Bengio, Hinton, LeCun for Deep Learning.

39. What are the Types of Machine Learning Relevant to NLP?

- **Machine Learning:** Acquiring models from data/experience. Learning parameters, structure, hidden concepts.
- **Types:**
    - **Unsupervised Learning:** No teacher feedback; detect patterns (e.g., clustering).
    - **Reinforcement Learning:** Feedback via rewards/punishments.
    - **Supervised Learning:** Given examples of correct input-output pairs (labeled data).
        - **Classification:** Discrete outputs (e.g., spam/ham, topic category, sentiment).
        - **Regression:** Continuous outputs.

40. How does Supervised Learning Work (e.g., Classification)?

- **Goal:** Given training set `(x1, y1), ..., (xn, yn)` where `yi = f(xi)` for unknown `f`, discover a hypothesis `h` that approximates `f`.
- **Examples:**
  - **Spam Filter:** Input `x`=email, Output `y`="spam" or "ham". Features: words (FREE!), text patterns ($dd), non-text (SenderInContacts).
  - **Digit Recognition:** Input `x`=image pixels, Output `y`=digit 0-9. Features: pixels, shape patterns (loops, aspect ratio).
- **Process:**
  1. **Data:** Collect labeled instances. Split into Training, Held-Out (Validation), and Test sets.
  2. **Features:** Define attributes characterizing `x`.
  3. **Experimentation Cycle:**
     - Learn model parameters (e.g., probabilities) on Training set.
     - Tune hyperparameters (e.g., learning rate, regularization) on Held-Out set.
     - Evaluate final performance (e.g., Accuracy) on Test set.
     - **Crucial:** Never "peek" at the Test set during training/tuning.
- **Evaluation:** Accuracy (fraction correct), Precision, Recall, F1-score.
- **Challenge: Overfitting:** Fitting training data too closely, failing to generalize to unseen test data. Need models that generalize well.

41. How is Text Classification used in NLP?

- **Task:** Assigning predefined labels/categories to documents.
- **Examples:**
  - Topic Labeling (Yahoo categories: finance, sports).
  - Genre Classification (editorials, reviews).
  - Opinion/Sentiment Analysis (like, hate, neutral).
  - Domain Specificity (interesting/not interesting).
  - Language Identification.
  - Spam Detection (email spam, link spam).
- **Methods History:**
  - **Manual Classification:** Accurate with experts, small scale. Expensive, hard to scale (Original Yahoo Directory, ODP).
  - **Hand-coded Rules:** Boolean combinations of words, standing queries. Accurate if refined by experts. Expensive to build/maintain (Reuters, gov agencies).
  - **Supervised Machine Learning:** Learns function from labeled data. Requires training data but can be built by amateurs. Widely used (k-NN, Naive Bayes, SVMs, Deep Learning). Many

commercial systems use hybrid approaches.

42. Why has Deep Learning become prominent in NLP?

Deep Learning is transforming the way machines understand, learn and interact with complex data. Deep learning mimics neural networks of the human brain, it enables computers to autonomously uncover patterns and make informed decisions from vast amounts of unstructured data.

- **Limitations of Shallow ML:** Required hand-crafted features (time-consuming, often incomplete/over-specified), suffered from curse of dimensionality with joint language models.
- **Deep Learning (DL) Advantages:**
  - Learns representations (features) *automatically* from data.
  - Effective at learning complex patterns through multiple layers of representation.
  - Flexible, (almost) universal framework.
  - Can learn supervised and unsupervised.
  - Effective end-to-end learning (raw input to final output).
  - Can leverage large amounts of training data.
  - Started outperforming other ML in ~2010 (Speech, Vision, then NLP).
- **Key DL Developments in NLP:**
  - Distributed Representations (Word Embeddings: Word2vec, GloVe; Contextual: ELMo, BERT).
  - Neural Architectures (CNNs, RNNs/LSTMs/GRUs, Recursive NN, Transformers).
  - Reinforcement Learning applications.
  - Unsupervised sentence representation learning.
  - Memory-augmented networks.
  - Attention mechanisms.
- **Impact:** Led to state-of-the-art results on many NLP tasks.

43. What is Stemming?

Stemming is a process in Natural Language Processing (NLP) used to reduce inflected (or sometimes derived) words to their word stem, base, or root form. The "stem" isn't necessarily a linguistically correct root word (like a lemma); it's often just a truncated version of the word.
Examples:

```
"running" -> "run"

"studies" -> "studi" (Note: "studi" is not a real word, but a common output of stemmers like Porter)

"studying" -> "studi"
```

`"beautiful" -> "beauti"`

`"connection," "connects," "connecting" -> "connect"`

Other Related NLP Preprocessing Processes:

Stemming is just one step in a typical text preprocessing pipeline. Here are other common processes:

1. **Tokenization:**
   - **What it is:** Breaking down a text (a sentence or a document) into smaller units called tokens. These tokens are usually words, but can also be punctuation marks, numbers, or sub-word units.
   - **Example:** "NLP is fascinating." -> `["NLP", "is", "fascinating", "."]`
2. **Lowercasing:**
   - **What it is:** Converting all text to lowercase.
   - **Purpose:** Ensures that words like "Apple" (the company) and "apple" (the fruit, or the company at the start of a sentence) are treated as the same token if context isn't crucial, reducing vocabulary size.
   - **Example:** "Apple" -> "apple"
3. **Stop Word Removal:**
   - **What it is:** Removing common words that occur frequently but usually don't carry significant meaning (e.g., "a," "an," "the," "is," "are," "in," "on").
   - **Purpose:** Reduces noise and dimensionality, allowing models to focus on more informative words.
   - **Example:** "This is a sentence." -> `["This", "sentence"]` (after removing "is", "a")
4. **Punctuation Removal/Handling:**
   - **What it is:** Removing or replacing punctuation marks.
   - **Purpose:** Simplifies the text, as punctuation often doesn't add value for many NLP tasks (though it can be important for sentiment analysis or syntax).
   - **Example:** "Hello, world!" -> `["Hello", "world"]`
5. **Lemmatization:**
   - **What it is:** Similar to stemming, it reduces words to their base or dictionary form, known as the **lemma**. Unlike stemming, lemmatization considers the word's meaning and part of speech. It uses a vocabulary and morphological analysis.
   - **Purpose:** More accurate normalization than stemming, as it produces actual dictionary words.
   - **Examples:**
     - "running" -> "run"
     - "studies" -> "study" (unlike stemming's "studi")

        - "better" -> "good" (if part-of-speech is known)
        - "mice" -> "mouse"
    - **Key Difference from Stemming:** Lemmatization is more linguistically informed and aims for a true dictionary root, while stemming is a cruder heuristic chopping process. Lemmatization is generally slower and more computationally intensive.
6. **Part-of-Speech (POS) Tagging:**
    - **What it is:** Assigning a grammatical category (e.g., noun, verb, adjective, adverb) to each token.
    - **Purpose:** Essential for lemmatization (e.g., "saw" as a verb vs. "saw" as a noun) and for many downstream NLP tasks like named entity recognition, syntax parsing, and question answering.
    - **Example:** "The cat sat." -> `[("The", "DET"), ("cat", "NOUN"), ("sat", "VERB")]`
7. **Named Entity Recognition (NER):**
    - **What it is:** Identifying and categorizing named entities in text into predefined categories such as person names, organizations, locations, medical codes, time expressions, quantities, monetary values, percentages, etc.
    - **Purpose:** Extracts key information from text.
    - **Example:** "Apple Inc. is headquartered in Cupertino." -> `[("Apple Inc.", "ORG"), ("Cupertino", "LOC")]`
8. **Sentence Segmentation (or Sentence Boundary Disambiguation):**
    - **What it is:** Dividing a text into its constituent sentences.
    - **Purpose:** Many NLP tasks operate at the sentence level.
    - **Challenge:** Handling abbreviations (e.g., "Mr.", "U.S.A.") which can be mistaken for sentence endings.

MIN-MAX algorithm:

- The **min max algorithm in AI,** popularly known as the minimax, is a backtracking algorithm.
- It is used in decision making, game theory and artificial intelligence (AI).
- It is used to find the optimal move for a player, assuming that the opponent is also playing optimally.
- Popular two-player computer or online games like Chess, Tic-Tac-Toe, Checkers, Go, etc. use this algorithm.

A backtracking algorithm is used to find a solution to computational problems in such a way that a candidate is incrementally built towards a solution, one step at a time. And the candidate that fails to complete a solution is immediately abandoned.

How does it work?

In the **min max algorithm in AI,** there are two players, Maximiser and Minimiser. Both these players play the game as one tries to get the highest score possible or the maximum benefit while the opponent tries to get the lowest score or the minimum benefit.

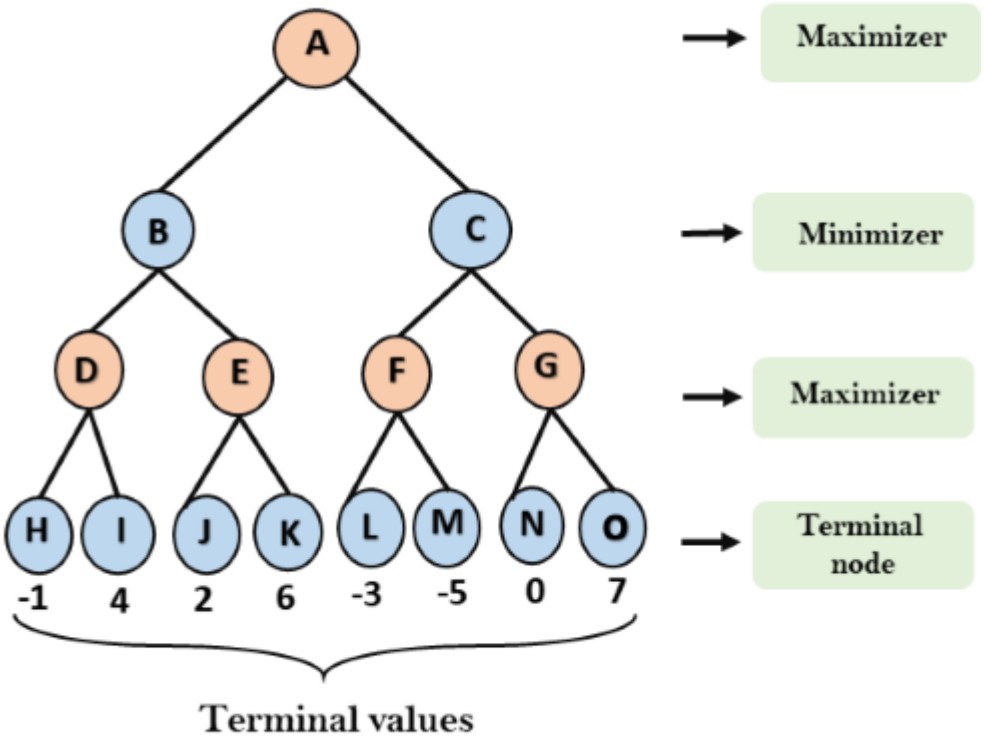
Every game board has an evaluation score assigned to it, so the Maximiser will select the maximised value, and the Minimiser will select the minimised value with counter moves. If the Maximiser has the upper hand, then the board score will be a positive value, and if the Minimiser has the upper hand, then the board score will be a negative value.

Working of Min-Max Algorithm:

- The working of the minimax algorithm can be easily described using an example. Below we have taken an example of game-tree which is representing the two-player game.
- In this example, there are two players one is called Maximizer and other is called Minimizer.

- Maximizer will try to get the Maximum possible score, and Minimizer will try to get the minimum possible score.
- This algorithm applies DFS, so in this game-tree, we have to go all the way through the leaves to reach the terminal nodes.
- At the terminal node, the terminal values are given so we will compare those value and backtrack the tree until the initial state occurs. Following are the main steps involved in solving the two-player game tree:

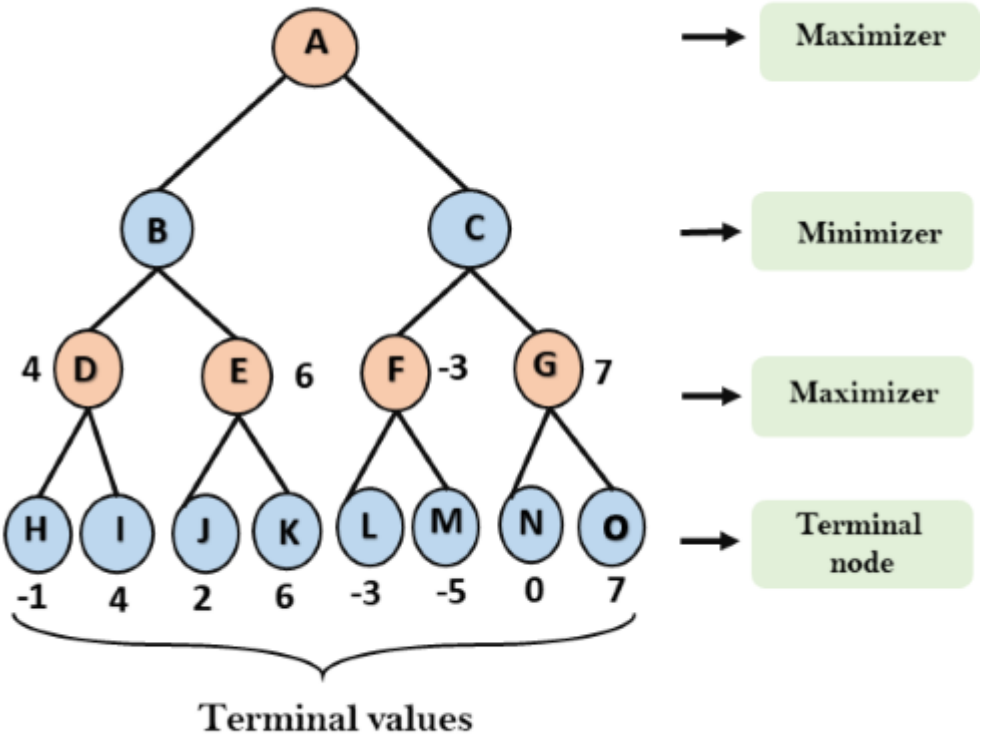
Step-1: In the first step, the algorithm generates the entire game-tree and apply the utility function to get the utility values for the terminal states. In the below tree diagram, let's take A is the initial state of the tree. Suppose maximizer takes first turn which has worst-case initial value =- infinity, and minimizer will take next turn which has worst-case initial value = +infinity.



Step 2: Now, first we find the utilities value for the Maximizer, its initial value is -∞, so we will compare each value in terminal state with initial value of Maximizer and determines the higher nodes values. It will find the maximum among the all.

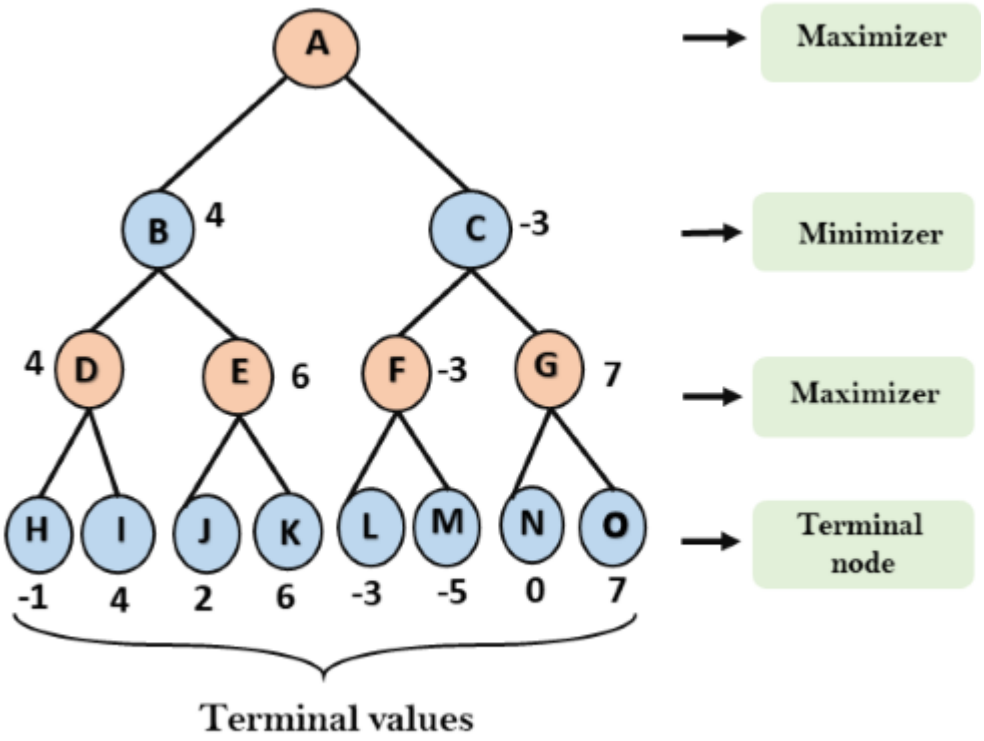
- For node D max(-1,- -∞) => max(-1,4)= 4

- For Node E max(2, -∞) => max(2, 6)= 6
- For Node F max(-3, -∞) => max(-3,-5) = -3
- For node G max(0, -∞) = max(0, 7) = 7



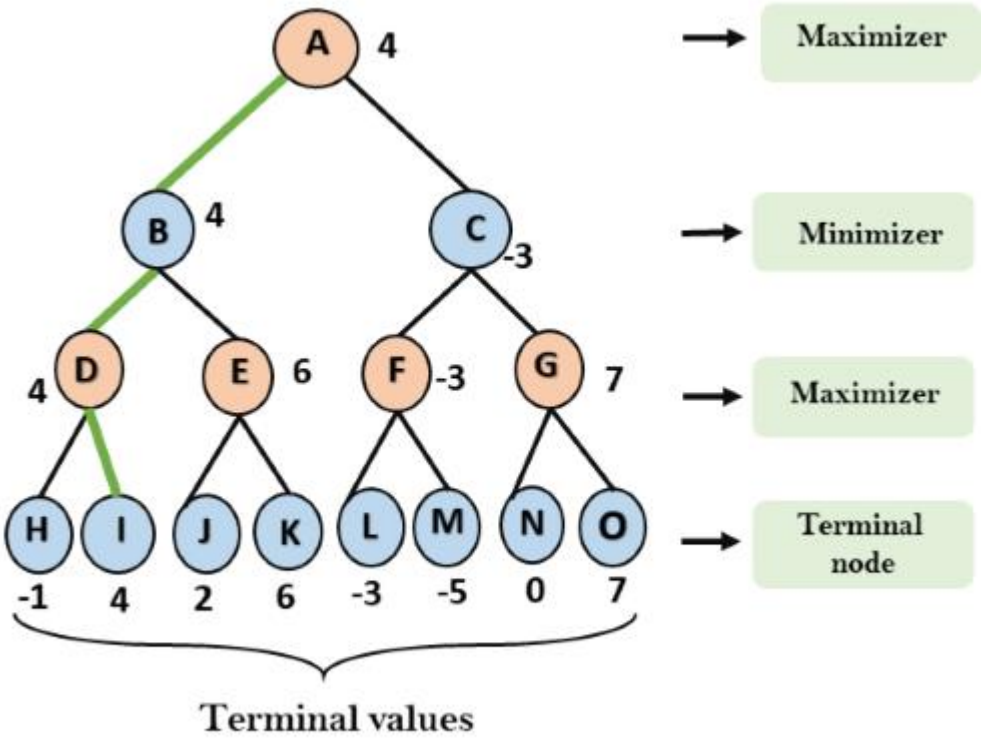
Step 3: In the next step, it's a turn for minimizer, so it will compare all nodes value with +∞, and will find the 3rd layer node values.

- For node B= min(4,6) = 4
- For node C= min (-3, 7) = -3



Step 4: Now it's a turn for Maximizer, and it will again choose the maximum of all nodes value and find the maximum value for the root node. In this game tree, there are only 4 layers, hence we reach immediately to the root node, but in real games, there will be more than 4 layers.

- For node A max(4, -3)= 4



That was the complete workflow of the minimax two player game.

Properties of Mini-Max algorithm:

- **Complete-** Min-Max algorithm is Complete. It will definitely find a solution (if exist), in the finite search tree.
- **Optimal-** Min-Max algorithm is optimal if both opponents are playing optimally.
- **Time complexity-** As it performs DFS for the game-tree, so the time complexity of Min-Max algorithm is **$O(b^m)$**, where b is branching factor of the game-tree, and m is the maximum depth of the tree.
- **Space Complexity-** Space complexity of Mini-max algorithm is also similar to DFS which is **O(bm)**.

Limitation of the minimax Algorithm:

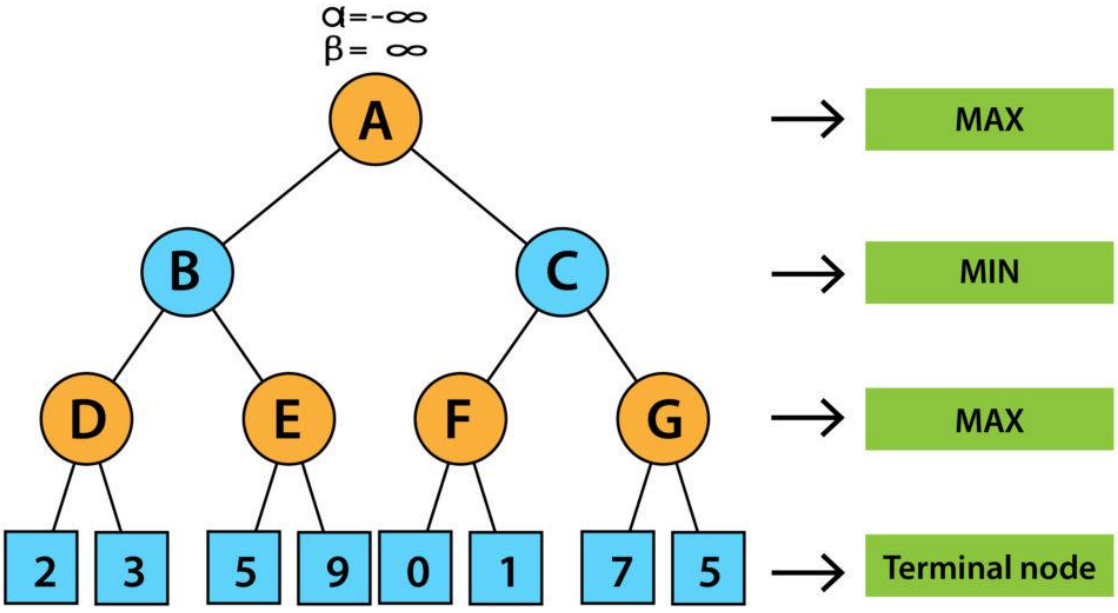
The main drawback of the minimax algorithm is that it gets really slow for complex games such as Chess, go, etc. This type of games has a huge branching factor, and the player has lots of choices to decide. This limitation of the minimax algorithm can be improved from **alpha-beta pruning** which we have discussed in the next topic.

Key points in Alpha-beta Pruning

- Alpha: Alpha is the best choice or the highest value that we have found at any instance along the path of Maximizer. The initial value for alpha is – ∞.
- Beta: Beta is the best choice or the lowest value that we have found at any instance along the path of Minimizer. The initial value for alpha is + ∞.
- The condition for Alpha-beta Pruning is that $\alpha >= \beta$.
- Each node has to keep track of its alpha and beta values. Alpha can be updated only when it's MAX's turn and, similarly, beta can be updated only when it's MIN's chance.
- MAX will update only alpha values and MIN player will update only beta values.
- The node values will be passed to upper nodes instead of values of alpha and beta during go into reverse of tree.
- Alpha and Beta values only be passed to child nodes.

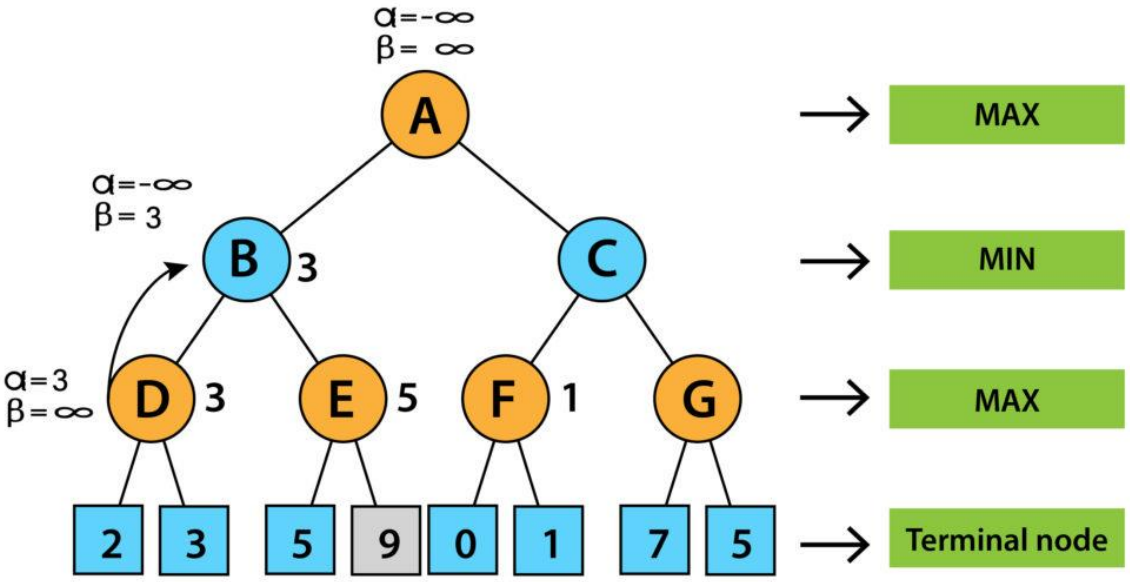
Working of Alpha-beta Pruning

1. We will first start with the initial move. We will initially define the alpha and beta values as the worst case i.e. $\alpha = -\infty$ and $\beta = +\infty$. We will prune the node only when alpha becomes greater than or equal to beta.



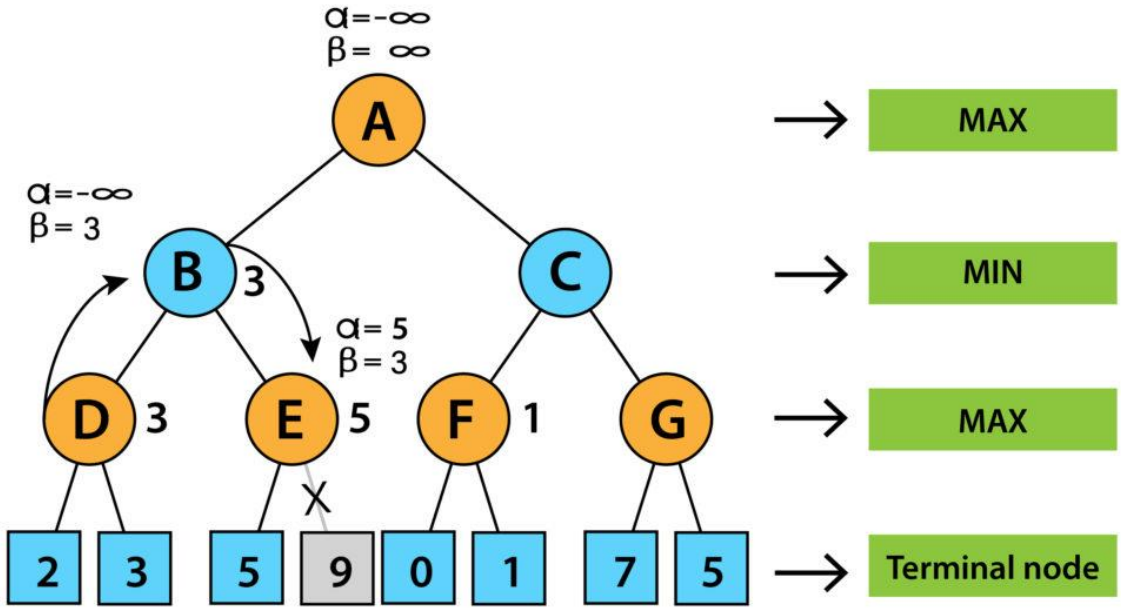
2. Since the initial value of alpha is less than beta so we didn't prune it. Now it's turn for MAX. So, at node D, value of alpha will be calculated. The value of alpha at node D will be max (2, 3). So, value of alpha at node D will be 3.

3. Now the next move will be on node B and its turn for MIN now. So, at node B, the value of alpha beta will be min (3, ∞). So, at node B values will be alpha= – ∞ and beta will be 3.



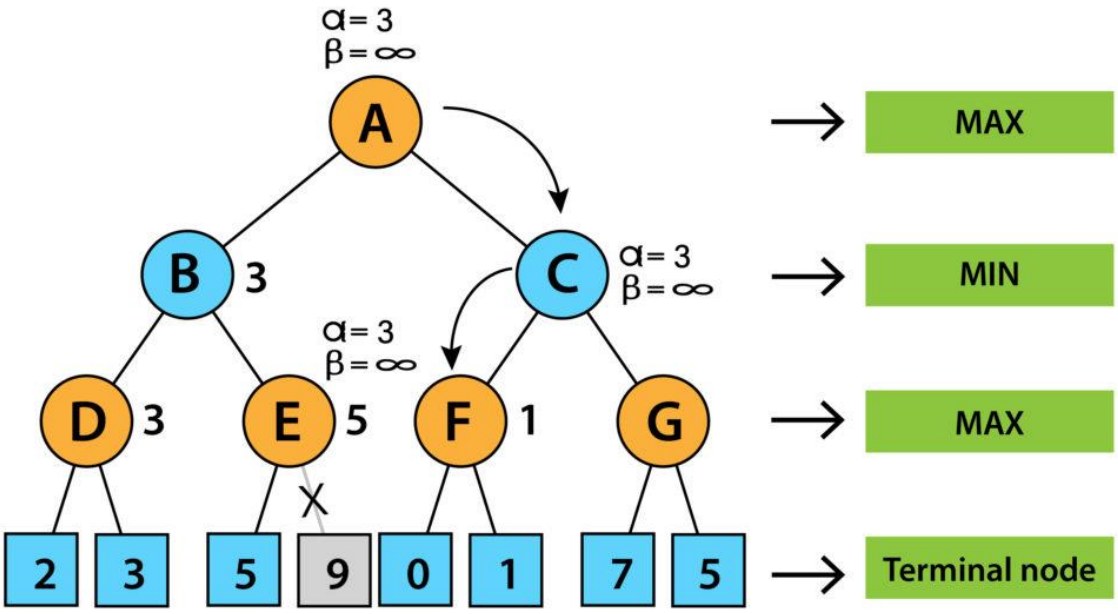
In the next step, algorithms traverse the next successor of Node B which is node E, and the values of α= -∞, and β= 3 will also be passed.

4. Now it's turn for MAX. So, at node E we will look for MAX. The current value of alpha at E is – ∞ and it will be compared with 5. So, MAX (- ∞, 5) will be 5. So, at node E, alpha = 5, Beta = 5. Now as we can see that alpha is greater than beta which is satisfying the pruning condition so we can prune the right successor of node E and algorithm will not be traversed and the value at node E will be 5.

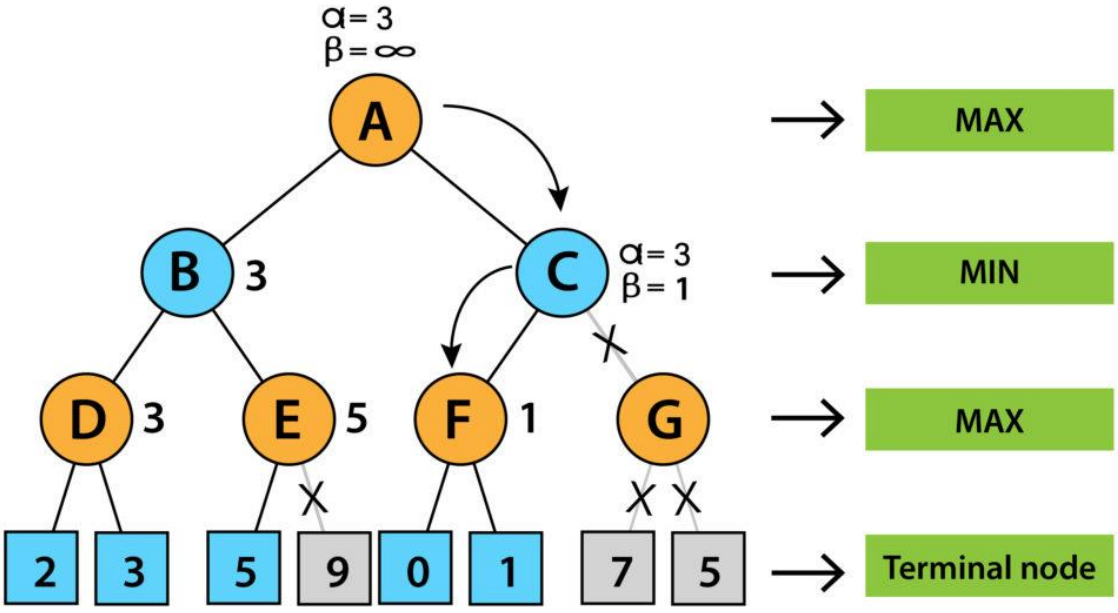


6. In the next step the algorithm again comes to node A from node B. At node A alpha will be changed to maximum value as MAX (- ∞, 3). So now the value of alpha and beta at node A will be (3, + ∞) respectively and will be transferred to node C. These same values will be transferred to node F.

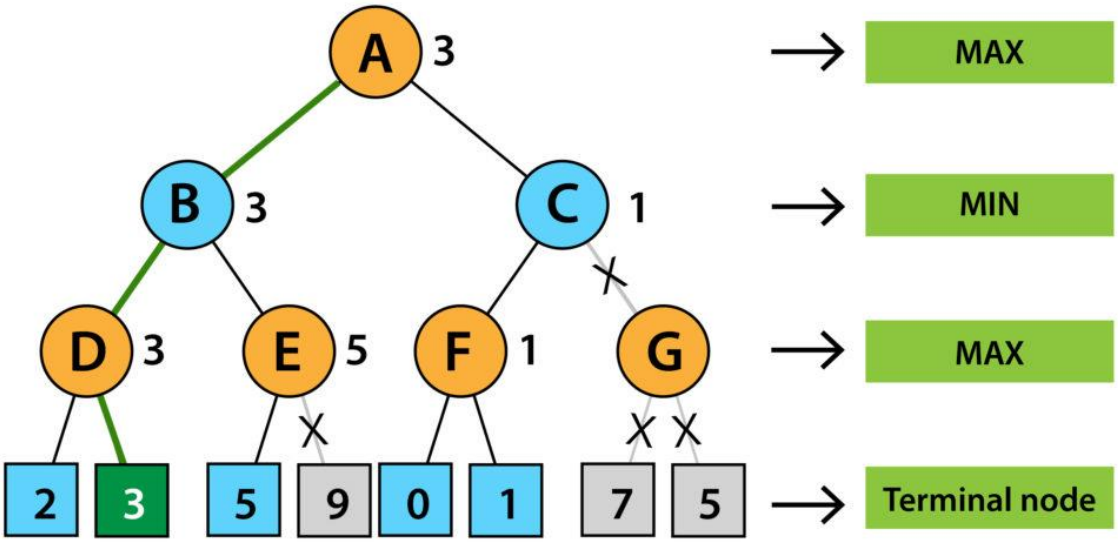
7. At node F the value of alpha will be compared to the left branch which is 0. So, MAX (0, 3) will be 3 and then compared with the right child which is 1, and MAX (3,1) = 3 still α remains 3, but the node value of F will become 1.



8. Now node F will return the node value 1 to C and will compare to beta value at C. Now its turn for MIN. So, MIN (+ ∞, 1) will be 1. Now at node C, α= 3, and β= 1 and alpha is greater than beta which again satisfies the pruning condition. So, the next successor of node C i.e. G will be pruned and the algorithm didn't compute the entire subtree G.



Now, C will return the node value to A and the best value of A will be MAX (1, 3) will be 3.



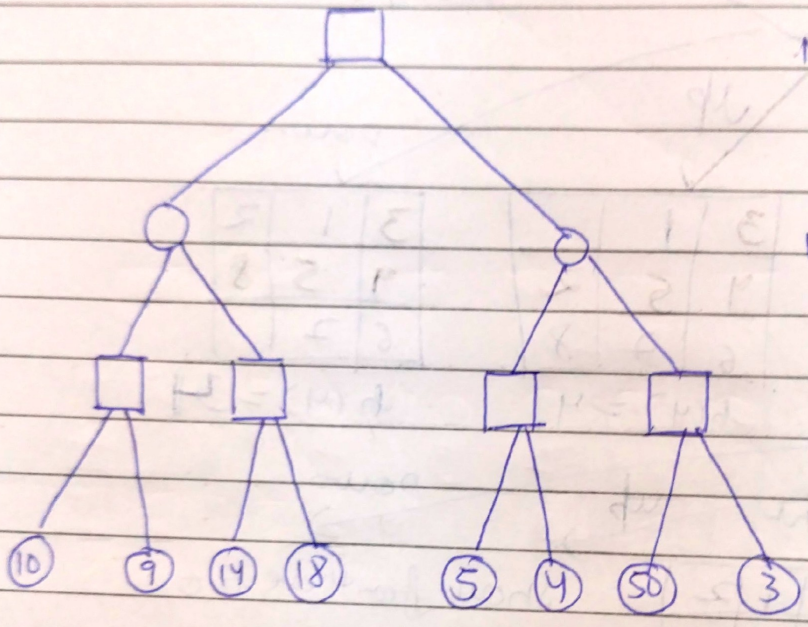
The above represented tree is the final tree which is showing the nodes which are computed and the nodes which are not computed. So, for this example the optimal value of the maximizer will be 3.

Game Playing

Mini Max Search Algorithm

Consider the following two player game tree in which the static scores are given from the first player's point of view

Ex (1)



⇒ Apply the Mini Max search algorithm and compute the value of the root of the tree

(Note: First player is Max in this case)

⇒ Also find the most convenient path for the Max Node.

Steps for Mini Max Search Algo.

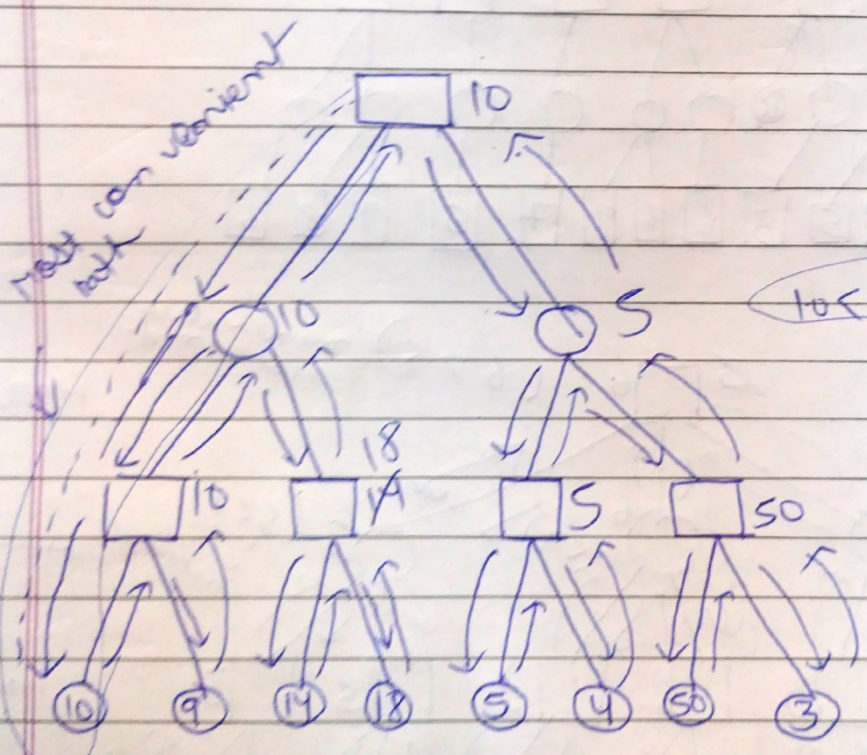
1. Generate the whole game tree to leaves
2. Apply utility (payoff) function to leaves.
3. Uses DFS for expanding the tree

④ Back up values from leaves towards the root.
- a Max node computes the maximum value from its child values.

- a Min node computes the minimum value from its child values.

⑤ When value reaches the root: optimal ~~value~~ move is determined.

Note: As in this question, tree is already given and values at the leaves are already given, Step ① & ② can be skipped.



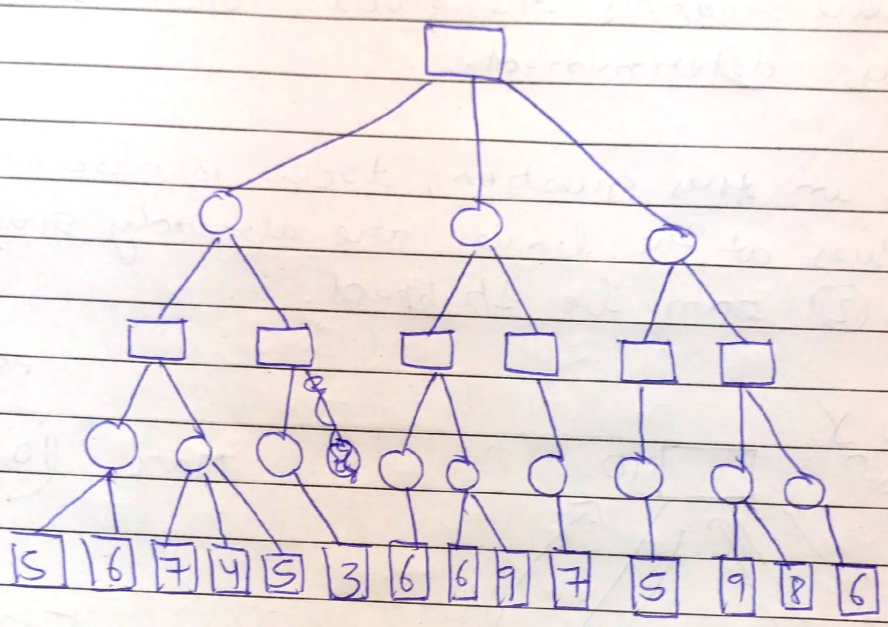
10>9 , 18>14 , 5>4 , 50>3 →

Now the value of root node = 10
Now check the path ~~ve~~ which has all the ~~interm~~ values as 10. It is the most convenient path.

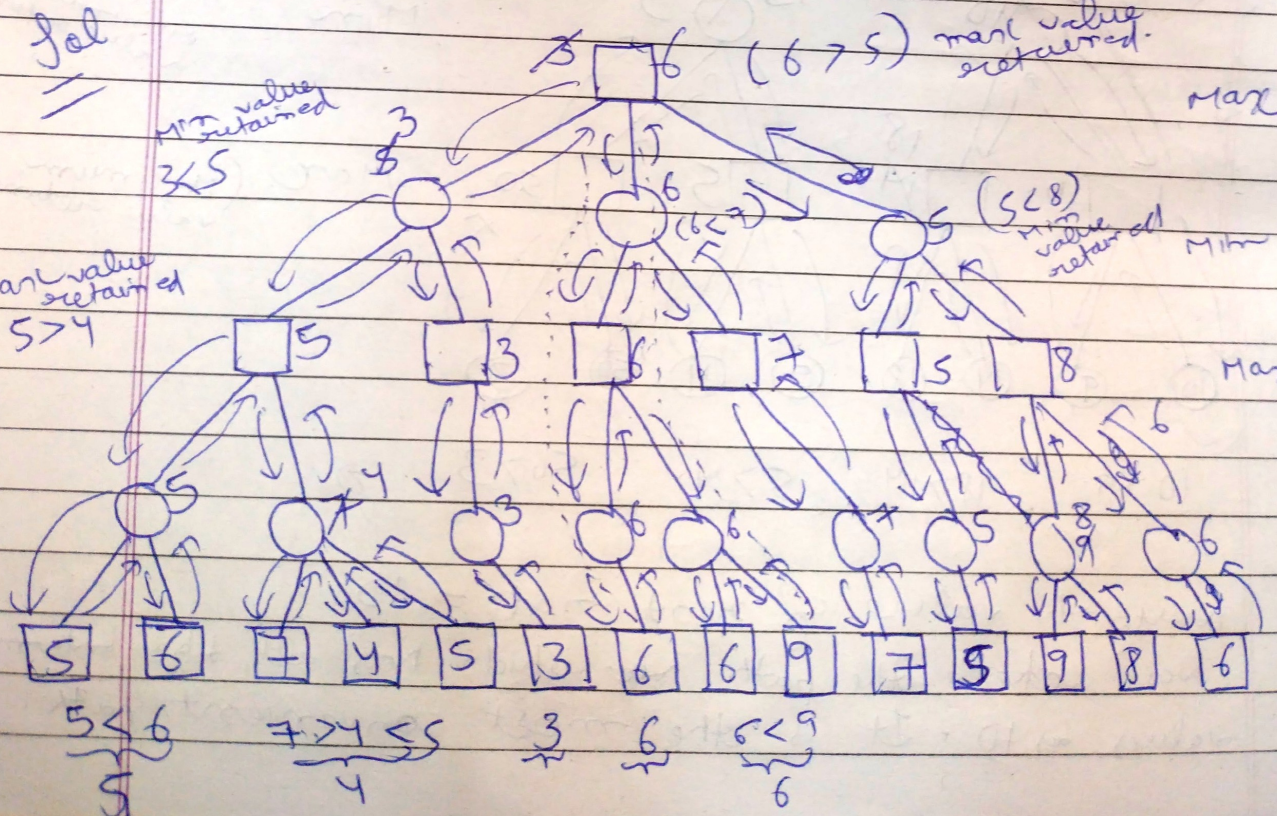
Page :
Date :

Ex 2 Consider the following two player game tree in which static scores are given from the first players point of view. □ max ○ min

Apply the Mini Max search algorithm and compute the value of the root of the tree. Also find the most convenient path for Max node.



Sol



Page :
Date :

Two The value of root node for max player to win is coming out to be 6.

Also two convenient paths have been identified shown as dotted lines which for can cause this winning.