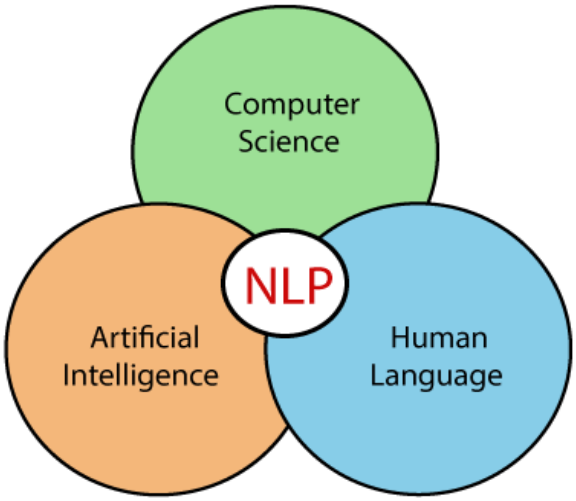
Topic – Natural language processing

Objective –

Outcome -

What is NLP?

NLP stands for **Natural Language Processing**, which is a part of **Computer Science, Human language,** and **Artificial Intelligence**. It is the technology that is used by machines to understand, analyse, manipulate, and interpret human's languages. It helps developers to organize knowledge for performing tasks such as **translation, automatic summarization, Named Entity Recognition (NER), speech recognition, relationship extraction,** and **topic segmentation**.



Advantages of NLP

- NLP helps users to ask questions about any subject and get a direct response within seconds.
- NLP offers exact answers to the question means it does not offer unnecessary and unwanted information.
- NLP helps computers to communicate with humans in their languages.
- It is very time efficient.
- Most of the companies use NLP to improve the efficiency of documentation processes, accuracy of documentation, and identify the information from large databases.

Disadvantages of NLP

A list of disadvantages of NLP is given below:

- NLP may not show context.
- NLP is unpredictable
- NLP may require more keystrokes.
- NLP is unable to adapt to the new domain, and it has a limited function that's why NLP is built for a single and specific task only.

Components of NLP

There are the following two components of NLP -

1. Natural Language Understanding (NLU)

Natural Language Understanding (NLU) helps the machine to understand and analyse human language by extracting the metadata from content such as concepts, entities, keywords, emotion, relations, and semantic roles.

NLU mainly used in Business applications to understand the customer's problem in both spoken and written language.

NLU involves the following tasks -

- It is used to map the given input into useful representation.
- It is used to analyze different aspects of the language.

2. Natural Language Generation (NLG)

Natural Language Generation (NLG) acts as a translator that converts the computerized data into natural language representation. It mainly involves Text planning, Sentence planning, and Text Realization.

Difference between NLU and NLG

NLU	NLG
NLU is the process of reading and interpreting language.	NLG is the process of writing or generating language.
It produces non-linguistic outputs from natural language inputs.	It produces constructing natural language outputs from non-linguistic inputs.

Difficulties in NLU

NL has an extremely rich form and structure.

It is very ambiguous. There can be different levels of ambiguity −

- **Lexical ambiguity** − It is at very primitive level such as word-level.

  For example, treating the word "board" as noun or verb?

- **Syntax Level ambiguity** − A sentence can be parsed in different ways.

  For example, "He lifted the beetle with red cap." − Did he use cap to lift the beetle or he lifted a beetle that had red cap?

- **Referential ambiguity** − Referring to something using pronouns. For example, Rima went to Gauri. She said, "I am tired." − Exactly who is tired?
- One input can mean different meanings.
- Many inputs can mean the same thing.

NLP Terminology

- **Phonology** − It is study of organizing sound systematically.
- **Morphology** − It is a study of construction of words from primitive meaningful units.
- **Morpheme** − It is primitive unit of meaning in a language.
- **Syntax** − It refers to arranging words to make a sentence. It also involves determining the structural role of words in the sentence and in phrases.
- **Semantics** − It is concerned with the meaning of words and how to combine words into meaningful phrases and sentences.
- **Pragmatics** − It deals with using and understanding sentences in different situations and how the interpretation of the sentence is affected.
- **Discourse** − It deals with how the immediately preceding sentence can affect the interpretation of the next sentence.
- **World Knowledge** − It includes the general knowledge about the world.

Applications of NLP

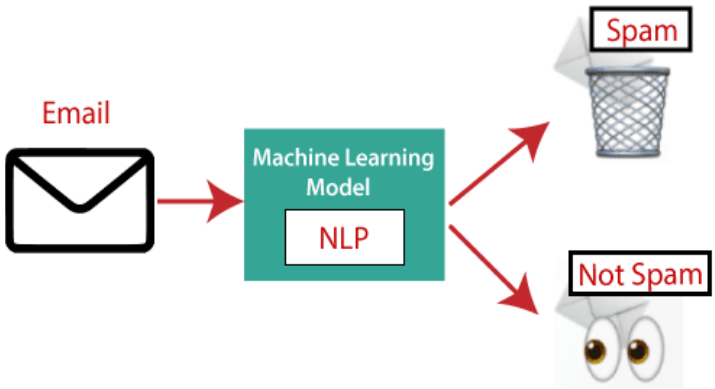
There are the following applications of NLP -

1. Question Answering

Question Answering focuses on building systems that automatically answer the questions asked by humans in a natural language.

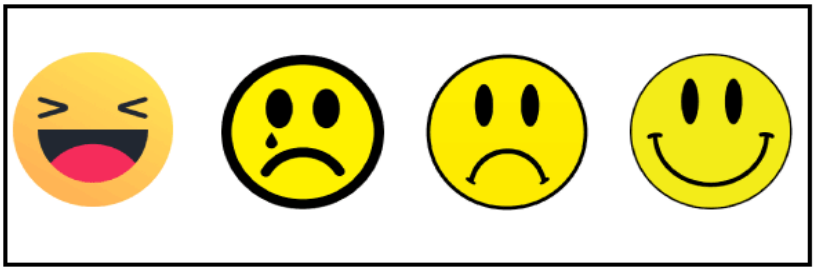
2. Spam Detection

Spam detection is used to detect unwanted e-mails getting to a user's inbox.



3. Sentiment Analysis

Sentiment Analysis is also known as **opinion mining**. It is used on the web to analyse the attitude, behaviour, and emotional state of the sender. This application is implemented through a combination of NLP (Natural Language Processing) and statistics by assigning the values to the text (positive, negative, or natural), identify the mood of the context (happy, sad, angry, etc.)



4. Machine Translation

Machine translation is used to translate text or speech from one natural language to another natural language.

5. Spelling correction

Microsoft Corporation provides word processor software like MS-word, PowerPoint for the spelling correction.

6. Speech Recognition

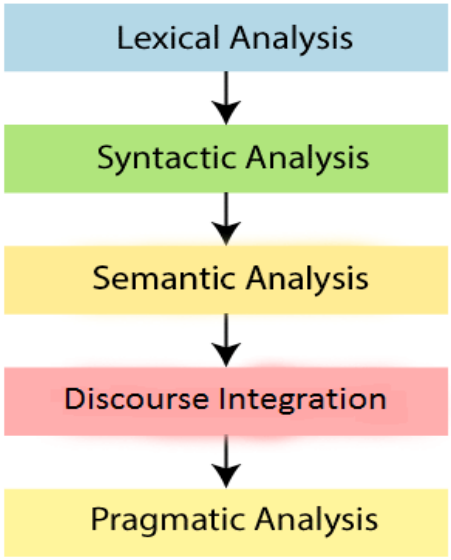
Speech recognition is used for converting spoken words into text. It is used in applications, such as mobile, home automation, video recovery, dictating to Microsoft Word, voice biometrics, voice user interface, and so on.

7. Chatbot

Implementing the Chatbot is one of the important applications of NLP. It is used by many companies to provide the customer's chat services.

Phases of NLP

There are the following five phases of NLP:



1. Lexical Analysis and Morphological

The first phase of NLP is the Lexical Analysis. This phase scans the source code as a stream of characters and converts it into meaningful lexemes. It divides the whole text into paragraphs, sentences, and words.

2. Syntactic Analysis (Parsing)

Syntactic Analysis is used to check grammar, word arrangements, and shows the relationship among the words.

Example: Agra goes to the Poonam

In the real world, Agra goes to the Poonam, does not make any sense, so this sentence is rejected by the Syntactic analyzer.

3. Semantic Analysis

Semantic analysis is concerned with the meaning representation. It mainly focuses on the literal meaning of words, phrases, and sentences.

4. Discourse Integration

Discourse Integration depends upon the sentences that proceeds it and also invokes the meaning of the sentences that follow it.

5. Pragmatic Analysis

Pragmatic is the fifth and last phase of NLP. It helps you to discover the intended effect by applying a set of rules that characterize cooperative dialogues.

For Example: "Open the door" is interpreted as a request instead of an order.

Summary -

Topic – Parsing techniques in AI

Objective –

Outcome –

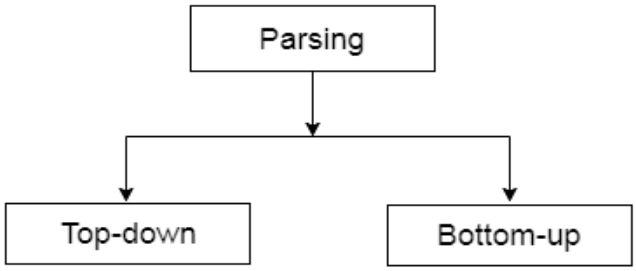
Parser

Parser is a compiler that is used to break the data into smaller elements coming from lexical analysis phase.

A parser takes input in the form of sequence of tokens and produces output in the form of parse tree.

Parsing is the term used to describe the process of automatically building syntactic analysis of a sentence in terms of a given grammar and lexicon. The resulting syntactic analysis may be used as input to a process of semantic interpretation. Occasionally, parsing is also used to include both syntactic and semantic analysis.

Parsing is of two types: top down parsing and bottom up parsing.



Top down parsing

- The top down parsing is known as recursive parsing or predictive parsing.
- Bottom up parsing is used to construct a parse tree for an input string.
- In the top down parsing, the parsing starts from the start symbol and transform it into the input symbol.

For parsing we will consider the previous symbols like PP, NP, VP, ART, N, V and so on. Examples of top down parsing are LL (Left-to-right, left most derivation), recursive descent parser etc.

Example 1: Rahul is eating an apple.

Symbolical Representation

S → NP VP

□ N VP (∴ NP □ N)

□ N AUX VP (□ VP □ AUX VP)

□ N AUX V NP (□ VP □ V NP)

□□ □UX V ART N (□□P □ ART N)

□□ □UX V ART apple

□ N AUX V an apple

□ N AUX eating an apple

□□ is eating an apple

□ Rahul is eating an apple.

Graphical Representation

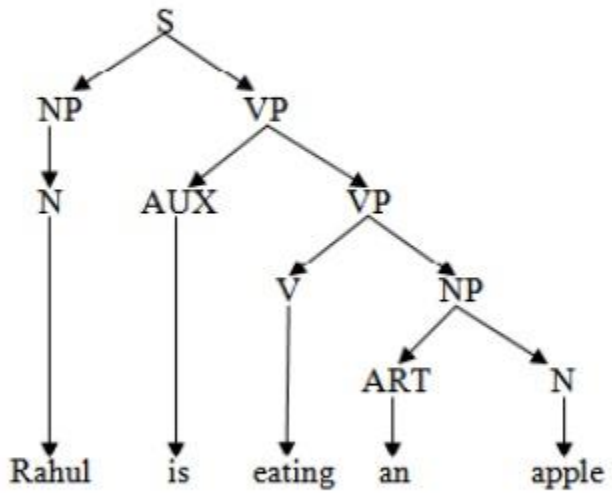


Figure Example of Top down Parsing

Example 2: The small tree shades the new house by the stream.

Symbolical Representation

S □NP VP

□ ART NP VP

□ The ADJ N VP

□ The small N V NP

□ The small tree V ART NP

□ The small tree shades ART ADJ NP

□ The small tree shades the ADJ N NP

□ The small tree shades the new N PREP N

□ The small tree shades the new house PREP ART N

□ The small tree shades the new house by ART N

□ The small tree shades the new house by the N

□ The small tree shades the new house by the stream.

Graphical Representation

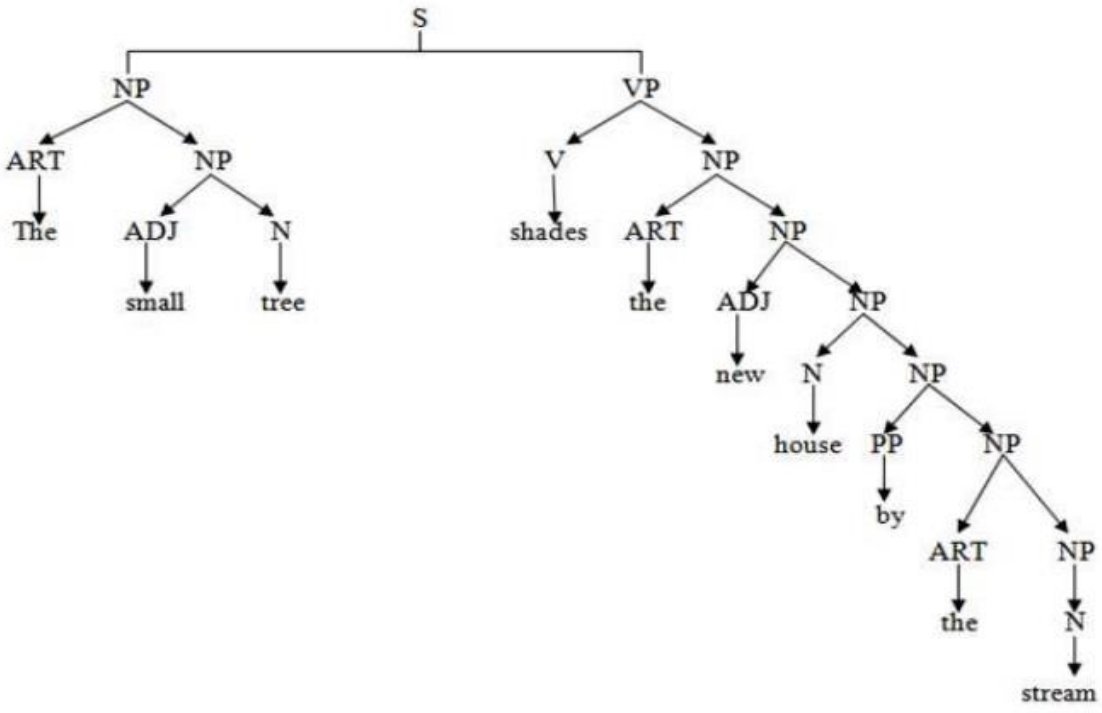


Figure Top down Parsing

Bottom up parsing

- Bottom up parsing is also known as shift-reduce parsing.
- Bottom up parsing is used to construct a parse tree for an input string.
- In the bottom up parsing, the parsing starts with the input symbol and construct the parse tree up to the start symbol by tracing out the rightmost derivations of string in reverse.
- Let us see some examples to illustrate the bottom up parsing.

Example-1:	Rahul	is	eating	an	apple.
	□□	is	eating	an	apple.
	□N	AUX	eating	an	apple.
	□N	AUX	V	an	apple.
	□N	AUX	V	ART	apple.
	□N	AUX	V	ART	N
	□N	AUX	V	NP	
	□N	AUX	VP		
	□N	VP			
	□NP	VP			
	□S				

Graphical Representation

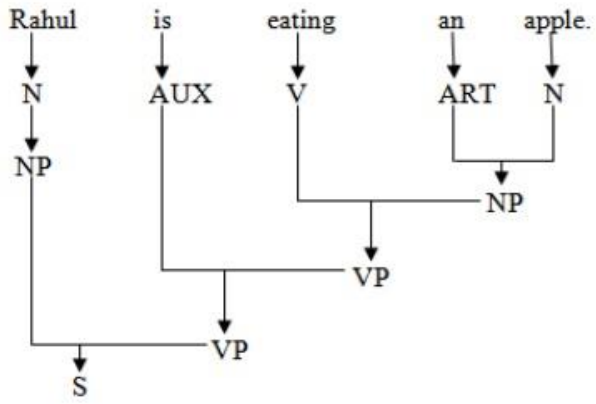


Figure Examples of Bottom up Parsing

Example-2:

□The small tree shades the new house by the stream

□ART small tree shades the new house by the stream

□ART ADJ tree shades the new house by the stream

□ART ADJ N shades the new house by the stream

□ART ADJ N V the new house by the stream

□ART ADJ N V ART new house by the stream

□ART ADJ N V ART ADJ house by the stream

□ART ADJ N V ART ADJ N by the stream

□ART ADJ N V ART ADJ N PREP the stream

□ART ADJ N V ART ADJ N PREP ART stream

□ART ADJ N V ART ADJ N PREP ART N

□ART ADJ N V ART ADJ N PREP NP

□ART ADJ N V ART ADJ N PP

□ART ADJ N V ART ADJ N PP

□ART ADJ N V ART ADJ NP

□ART ADJ N V ART NP

□ART ADJ N V NP

□ART ADJ N VP

□ART NP VP

□NP VP

□S

Graphical Representation

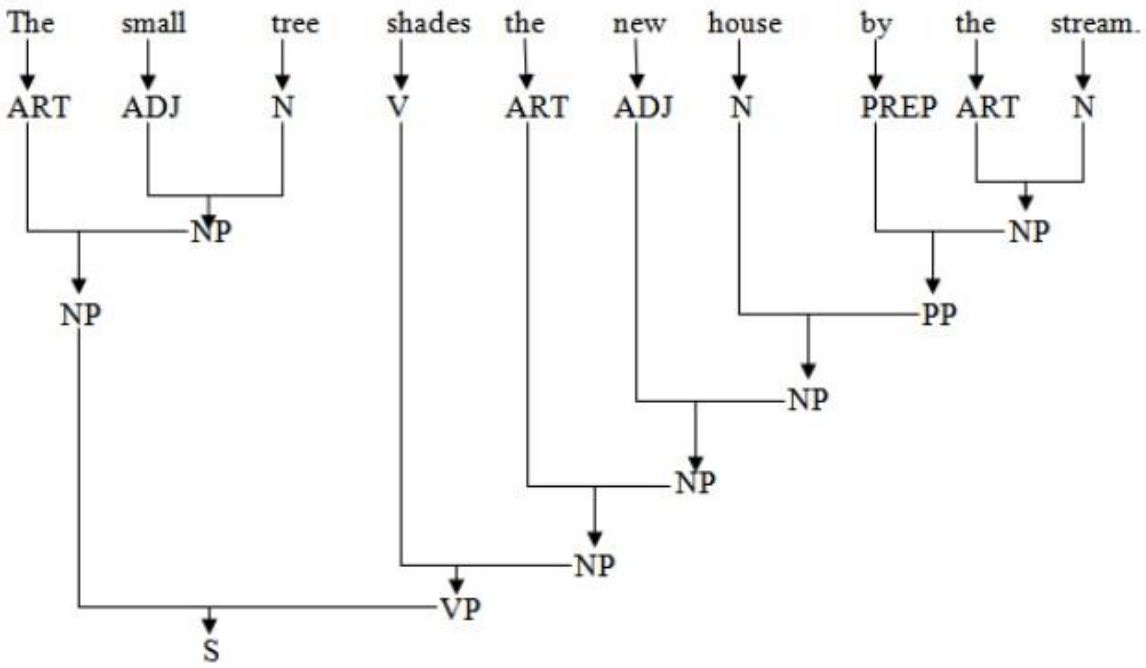


Figure Example of Bottom up Parsing

Summary -

Topic – Game playing, Minimax search procedure, Alpha-beta cutoff

Objective –

Outcome –

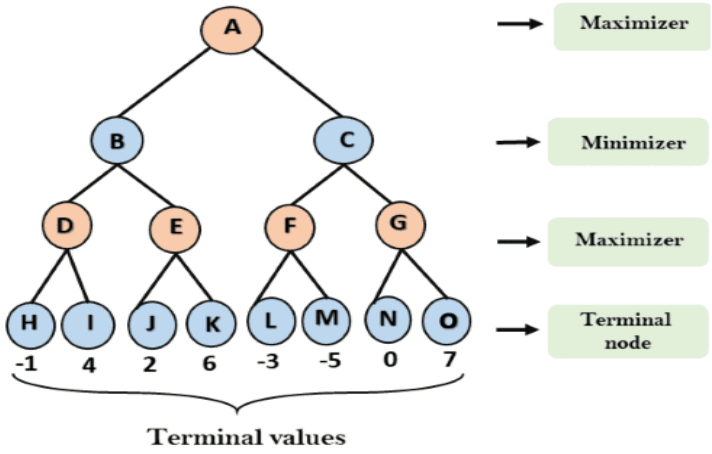
Mini-Max Algorithm in Artificial Intelligence

- In decision-making and game theory, the mini-max algorithm is a recursive or backtracking method. It suggests the best move for the player, provided that the opponent is likewise playing well.
- In AI, the Min-Max algorithm is mostly employed for game play. Chess, checkers, tic-tac-toe, go, and other two-player games are examples. This Algorithm calculates the current state's minimax choice.
- The game is played by two players, one named MAX and the other named MIN, in this algorithm.
- Both players FIGHT it, since the opponent player receives the smallest benefit while they receive the greatest profit.
- Both players in the game are adversaries, with MAX selecting the maximum value and MIN selecting the minimum value.
- For the exploration of the entire game tree, the minimax method uses a depth-first search strategy.
- For the exploration of the entire game tree, the minimax method uses a depth-first search strategy.
- The minimax algorithm descends all the way to the tree's terminal node, then recursively backtracks the tree.

Working of Min-Max Algorithm:

- A simple example can be used to explain how the minimax algorithm works. We've included an example of a game-tree below, which represents a two-player game.
- There are two players in this scenario, one named Maximizer and the other named Minimizer.
- Maximizer will strive for the highest possible score, while Minimizer will strive for the lowest possible score.
- Because this algorithm uses DFS, we must go all the way through the leaves to reach the terminal nodes in this game-tree.
- The terminal values are given at the terminal node, so we'll compare them and retrace the tree till we reach the original state. The essential phases in solving the two-player game tree are as follows:

Step 1: The method constructs the whole game-tree and applies the utility function to obtain utility values for the terminal states in the first step. Let's assume A is the tree's initial state in the diagram below. Assume that the maximizer takes the first turn with a worst-case initial value of -infinity, and the minimizer takes the second turn with a worst-case initial value of +infinity.

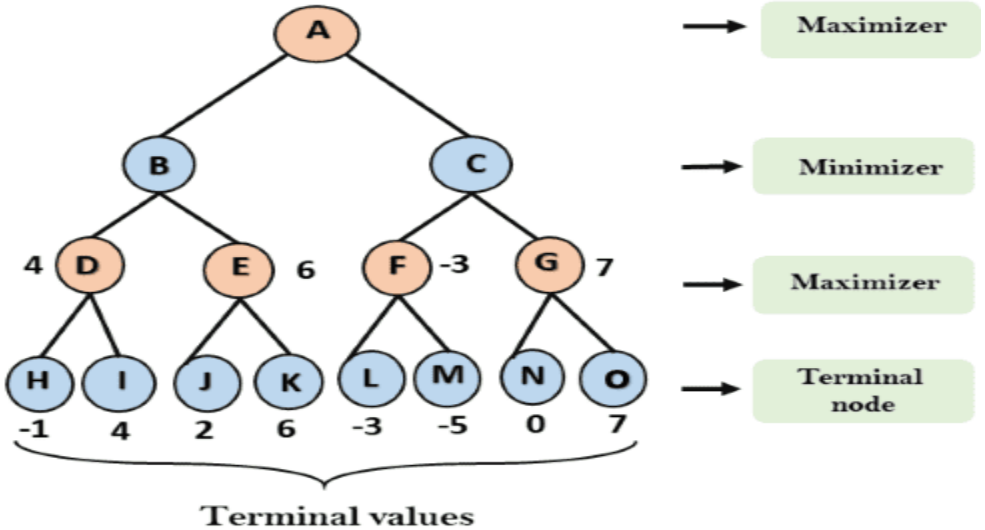


Example: Working of Min-Max Algorithm

step-1

Step 2: Next, we'll locate the Maximizer's utilities value, which is -, and compare each value in the terminal state to the Maximizer's initial value to determine the upper nodes' values. It will select the best option from all of them.

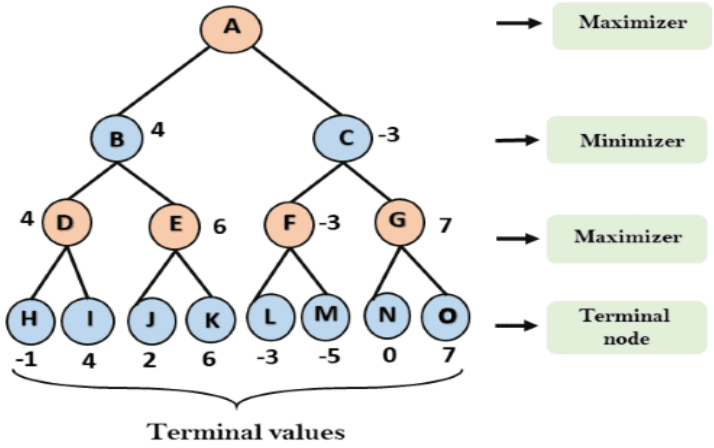
- **max(-1,- -∞) = For node D > max(-1,4)= 4**
- **max(2, -∞) = For Node E > max(2, 6)= 6**
- **max(-3, -∞) = For Node F > max(-3,-5) = -3**
- **max(0, -∞) = max(0, 7) = 7 For node G**



Step-2

Step 3: Now it's the minimizer's time, thus it'll compare all nodes' values with + and determine the 3rd layer node values.

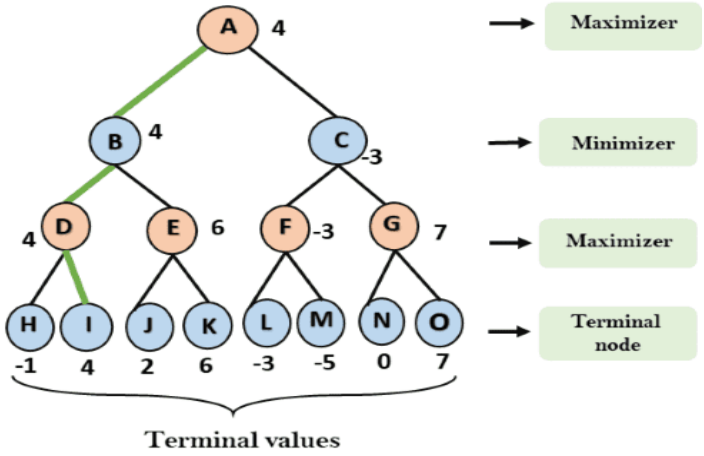
- **For node B = min(4,6) = 4**
- **For node C = min (-3, 7) = -3**



Step 3

In the next step, algorithm traverse the next successor of Node B which is node E, and the values of $\alpha = -\infty$, and $\beta = 3$ will also be passed.

Step 4: Now it's Maximizer's turn, and it'll choose the maximum value of all nodes and locate the root node's maximum value. There are only four layers in this game tree, so we can go to the root node right away, but there will be more layers in real games.
For node A max(4, -3)= 4



Step 4

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

Properties of Mini-Max algorithm:

- **Complete –**The Min-Max algorithm is finished. In the finite search tree, it will undoubtedly locate a solution (if one exists).
- **Optimal-** If both opponents are playing optimally, the Min-Max algorithm is optimal.
- **Time complexity-** Because it executes DFS for the game-tree, the time complexity of the Min-Max algorithm is **$O(b^m)$**, where b is the game-branching tree's factor and m is the tree's maximum depth.
- **Space Complexity-** Mini-max method has a space complexity that is similar to DFS, which is **O(bm)**.

Limitation of the minimax Algorithm:

The biggest disadvantage of the minimax algorithm is that it becomes extremely slow while playing complex games like chess or go. This style of game contains a lot of branching, and the player has a lot of options to choose from. The minimax algorithm's drawback can be alleviated by using **alpha-beta pruning** , which we will explore in the next section. the depth to which the tree can grow.

ALPHA-BETA PRUNING

- **A MODIFIED VARIANT OF** the minimax method is alpha-beta pruning. It's a way for improving the minimax algorithm.
- The number of game states that the minimax search algorithm must investigate grows exponentially with the depth of the tree, as we saw with the minimax search method. We can't get rid of the exponent, but we can reduce it by half. As a result, there is a technique known as **Pruning** that allows us to compute the correct minimax choice without having to inspect each node of the game tree. It's named **alpha-beta pruning** because it involves two threshold parameters, Alpha and beta, for future expansion. **Alpha-Beta Algorithm** is another name for it.
- Alpha-beta pruning can be done at any depth in a tree, and it can sometimes prune the entire sub-tree as well as the tree leaves.
- The two-parameter can be written as follows:
  a) **Alpha:** At any point along the Maximizer path, Alpha is the best (highest-value) option we've uncovered so far. The value of alpha starts at -.
  b) **Beta:** At every point along the route of Minimizer. Beta is the best (lowest-value) option we've found so far. Minimizer. The initial value of beta is $+\infty$
- The Alpha-beta pruning to a standard minimax algorithm produces the same result as the regular approach, but it removes those nodes that aren't really effecting the final decision but are slowing down the procedure. As a result, reducing these nodes speeds up the process.

Condition for Alpha-beta pruning:
The main condition which required for alpha-beta pruning is:

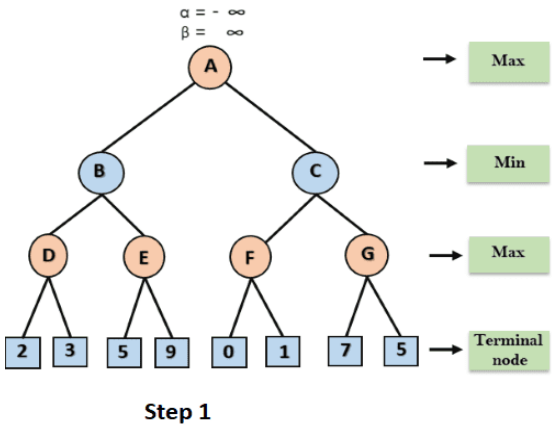
```
α >= β
```

Key points about alpha-beta pruning:

- Only the value of alpha will be updated by the Max player.
- Only the beta value will be updated by the Min player.
- Instead of alpha and beta values, node values will be sent to upper nodes while retracing the tree.
- Only the alpha and beta values will be passed to the child nodes.

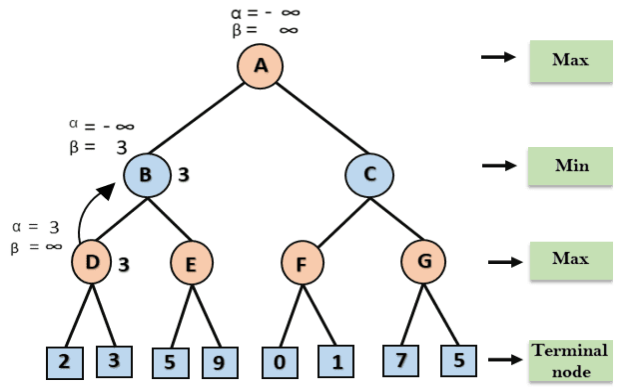
Working of Alpha-Beta Pruning:

- To better understand how Alpha-beta pruning works, consider a two-player search tree. **Step 1:** The Max player will begin by moving from node A, where = - and = +, and passing these values of alpha and beta to node B, where again = - and = +, and Node B passing the same value to its offspring D.



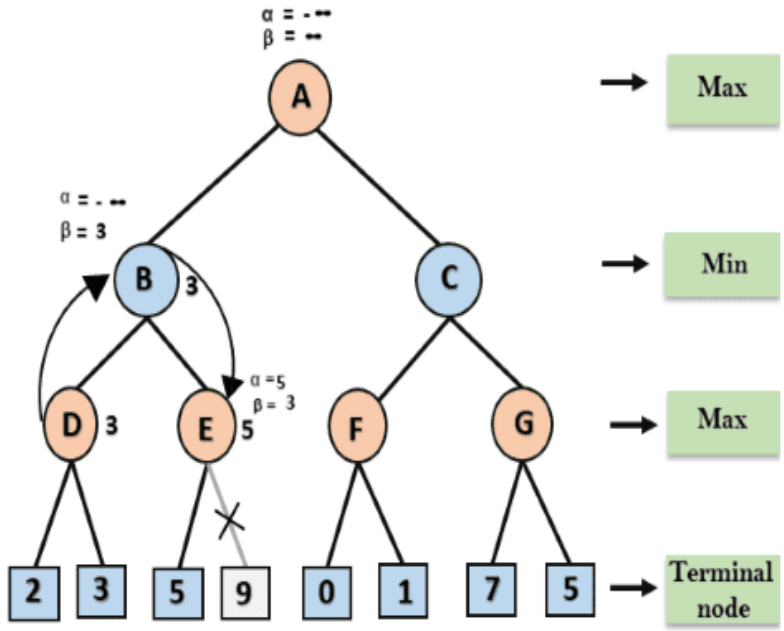
Step 1

- **Step 2:** The value of will be determined as Max's turn at Node D. The value of is compared to 2, then 3, and the value of at node D will be max (2, 3) = 3, and the node value will also be 3.
- **Step 3:** The algorithm now returns to node B, where the value of will change as this is a turn of Min, now = +, and will compare with the value of the available subsequent nodes, i.e. min (, 3) = 3, so at node B now = -, and= 3.



Step 3

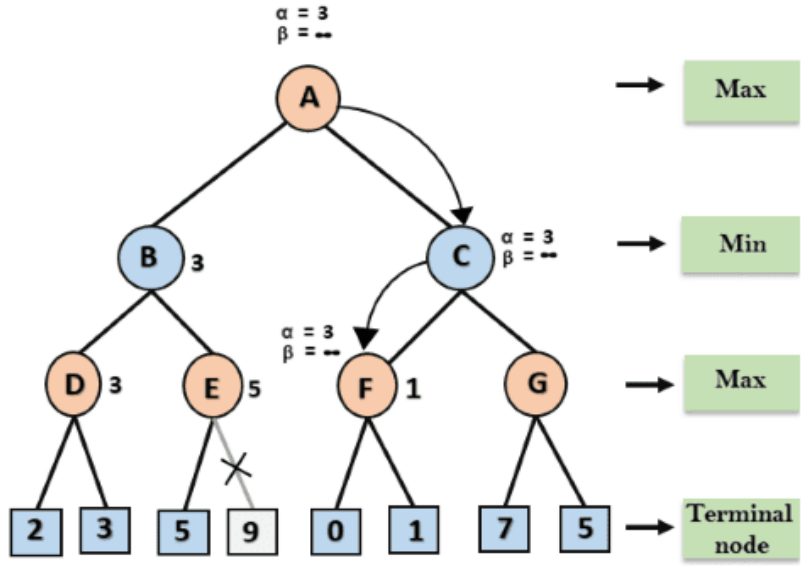
- In the next step, algorithm traverse the next successor of Node B which is node E, and the values of α= -∞, and β= 3 will also be passed.
- **Step 4:** Max will take its turn at node E, changing the value of alpha. The current value of alpha will be compared to 5, resulting in max (-, 5) = 5, and at node E= 5 and= 3, where >=, the right successor of E will be pruned, and the algorithm will not traverse it, and the value at node E will be 5.



Step 4

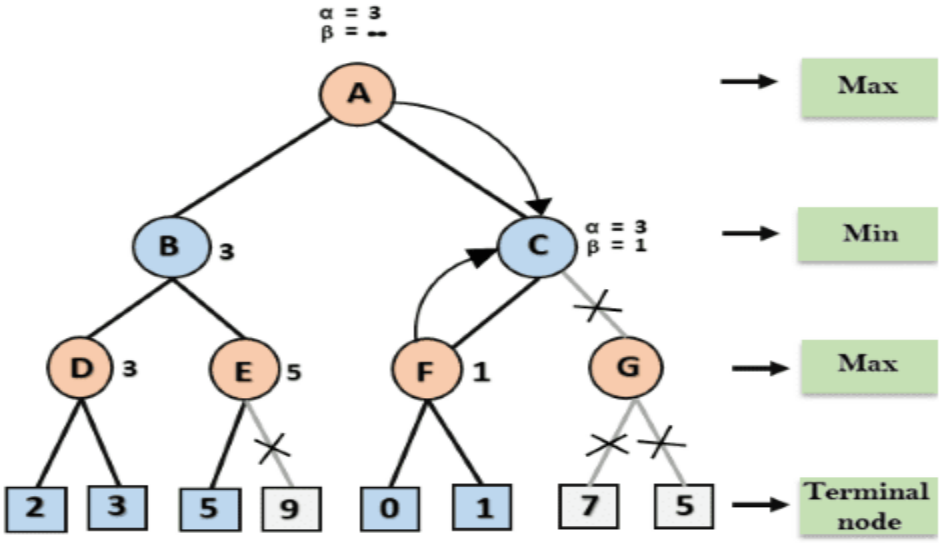
- **Step 5:** The method now goes backwards in the tree, from node B to node A. The value of alpha will be modified at node A, and the highest available value will be 3 as max (-, 3)= 3, and = +. These two values will now transfer to A's right successor, Node C.

  =3 and = + will be passed on to node F at node C, and the same values will be passed on to node F.
- **Step 6:** TAt node F, the value of will be compared with the left child, which is 0, and max(3,0)= 3, and then with the right child, which is 1, and max(3,1)= 3 will remain the same, but the node valuse of F will change to 1.



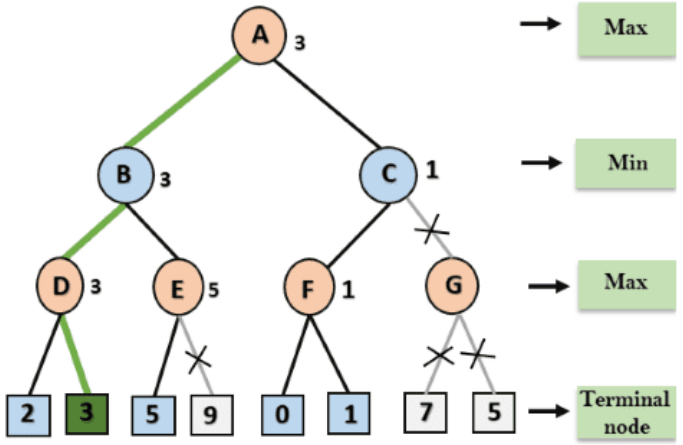
Step 6

- **Step 7:** Node F returns the node value 1 to node C, at C = 3 and = +, the value of beta is modified, and it is compared to 1, resulting in min (, 1) = 1. Now, at C, =3 and = 1, and again, it meets the condition >=, the algorithm will prune the next child of C, which is G, and will not compute the complete sub-tree G.



Step 7

- **Step 8:** C now returns 1 to A, with max (3, 1) = 3 being the greatest result for A. The completed game tree, which shows calculated and uncomputed nodes, is shown below. As a result, in this case, the best maximizer value is 3.



Step 8

Rules to find good ordering:

The following are some guidelines for finding effective alpha-beta pruning ordering:

- The optimal move should be made from the shallowest node.
- In the tree, sort the nodes so that the best ones are checked first.
- When deciding on the right step, make use of your subject knowledge. For example, in Chess, attempt the following order: captures first, threats second, forward moves third, backward moves fourth.
- We can keep track of the states because there's a chance they'll happen again.

Summary -

Topic – Game playing, Minimax search procedure, Alpha-beta cutoff

Objective –

Outcome –

What is planning in AI?

- The planning in Artificial Intelligence is about the decision making tasks performed by the robots or computer programs to achieve a specific goal.
- The execution of planning is about choosing a sequence of actions with a high likelihood to complete the specific task.

Blocks-World planning problem

- The blocks-world problem is known as **Sussman Anomaly.**
- Non interleaved planners of the early 1970s were unable to solve this problem, hence it is considered as anomalous.
- When two sub goals G1 and G2 are given, a non interleaved planner produces either a plan for G1 concatenated with a plan for G2, or vice-versa.
- In blocks-world problem, three blocks labeled as 'A', 'B', 'C' are allowed to rest on the flat surface. The given condition is that only one block can be moved at a time to achieve the goal.
- The start state and goal state are shown in the following diagram.

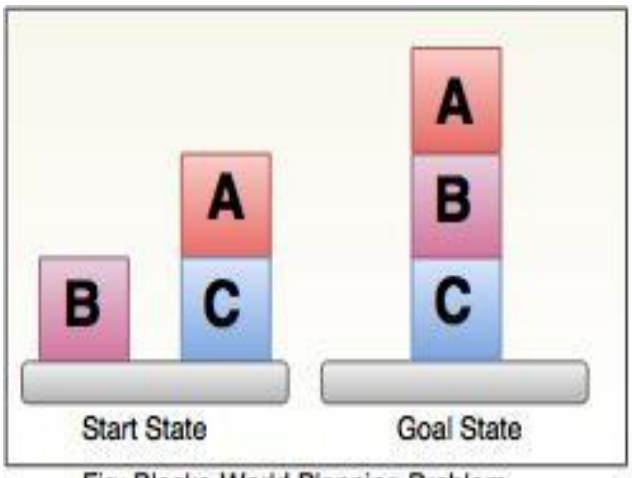


Fig: Blocks-World Planning Problem

Components of Planning System

The planning consists of following important steps:

- Choose the best rule for applying the next rule based on the best available heuristics.
- Apply the chosen rule for computing the new problem state.
- Detect when a solution has been found.
- Detect dead ends so that they can be abandoned and the system's effort is directed in more fruitful directions.
- Detect when an almost correct solution has been found.

Goal stack planning

This is one of the most important planning algorithms, which is specifically used by **STRIPS.**

- The stack is used in an algorithm to hold the action and satisfy the goal. A knowledge base is used to hold the current state, actions.
- Goal stack is similar to a node in a search tree, where the branches are created if there is a choice of an action.

The important steps of the algorithm are as stated below:

i. Start by pushing the original goal on the stack. Repeat this until the stack becomes empty. If stack top is a compound goal, then push its unsatisfied subgoals on the stack.
ii. If stack top is a single unsatisfied goal then, replace it by an action and push the action's precondition on the stack to satisfy the condition.
iii. If stack top is an action, pop it from the stack, execute it and change the knowledge base by the effects of the action.
iv. If stack top is a satisfied goal, pop it from the stack.

Non-linear planning

This planning is used to set a goal stack and is included in the search space of all possible subgoal orderings. It handles the goal interactions by interleaving method.

Advantage of non-Linear planning

Non-linear planning may be an optimal solution with respect to plan length (depending on search strategy used).

Disadvantages of Nonlinear planning

- It takes larger search space, since all possible goal orderings are taken into consideration.
- Complex algorithm to understand.

Algorithm

1. Choose a goal 'g' from the goalset
2. If 'g' does not match the state, then

- Choose an operator 'o' whose add-list matches goal g
- Push 'o' on the opstack
- Add the preconditions of 'o' to the goalset

3. While all preconditions of operator on top of opstack are met in state

- Pop operator o from top of opstack
- state = apply(o, state)
- plan = [plan; o]

Summary -