

Adversarial Search

- It is a **game-playing** technique where agents are surrounded by a competitive environment. A conflicting goal is given to the agents.
- Game-playing means discussing those games where **human intelligence & logic factor** is used. Tic-tac-toe, chess, checkers.
- This search is based on the concept of **'Game Theory.'**
- To complete game, one has to win the game & other loses automatically.



We are opponents- I win, you loose.

Techniques required to get the best optimal solution

There is always a need to choose those algorithms which provide the best optimal solution in a limited time. So, we use the following techniques which could fulfill our requirements:

- **Pruning:** A technique which allows ignoring the unwanted portions of a search tree which make no difference in its final result.
- **Heuristic Evaluation Function:** It allows to approximate the cost value at each level of the search tree, before reaching the goal node.

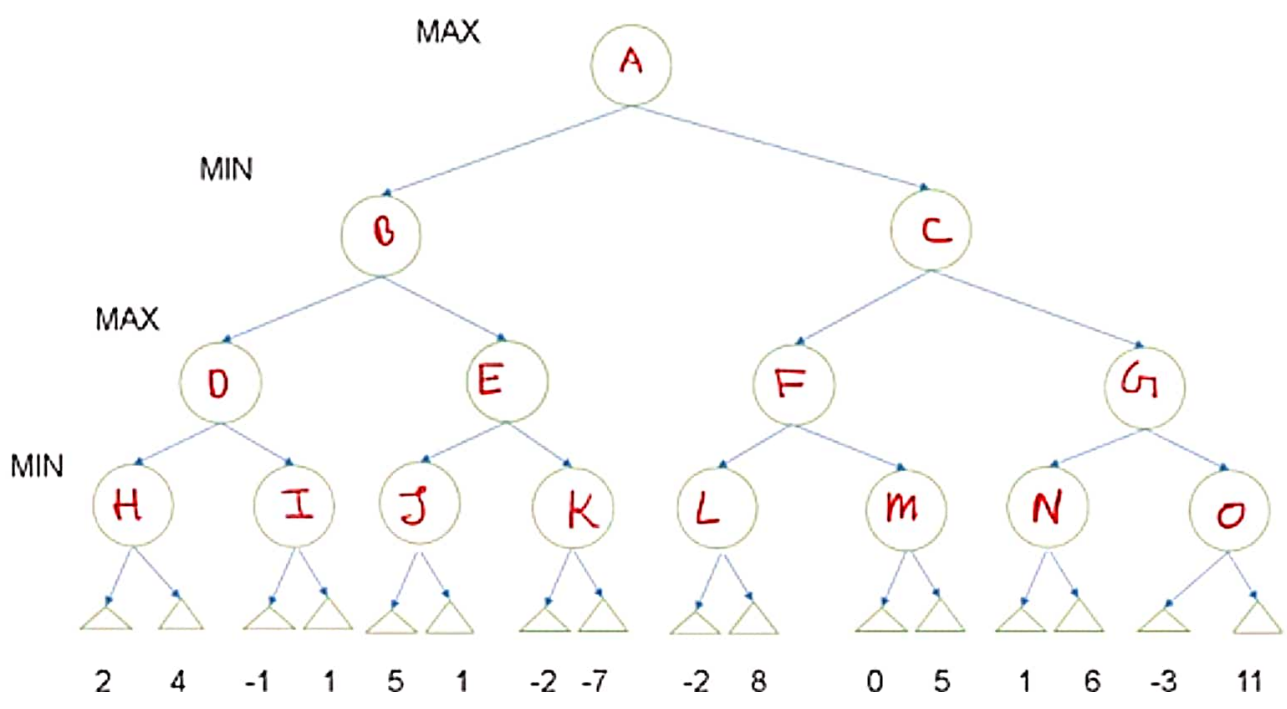
Types of algorithms in Adversarial search

- In a **normal search**, we follow a sequence of actions to reach the goal or to finish the game optimally. But in an **adversarial search**, the result depends on the players which will decide the result of the game.
- It is also obvious that the solution for the goal state will be an optimal solution because the player will try to win the game with the shortest path and under limited time.
- There are following types of adversarial search:
 - **Min-max Algorithm**
 - **Alpha-beta Pruning**

Min-Max Algorithm

- It is a recursive or backtracking algorithm which is used in decision-making and game theory. It provides an optimal move for the player assuming that opponent is also playing optimally.
- Mini-Max algorithm uses recursion to search through the game-tree.
- It is mostly used for game playing in AI. Such as Chess, Checkers, tic-tac-toe, go, and various tow-players game.
- In this algorithm two players play the game, one is called MAX and other is called MIN.
- Both the players fight it as the opponent player gets the minimum benefit while they get the maximum benefit.
- Both Players of the game are opponent of each other, where MAX will select the maximized value and MIN will select the minimized value.
- The minimax algorithm performs a depth-first search algorithm for the exploration of the complete game tree.

Min-Max Algorithm



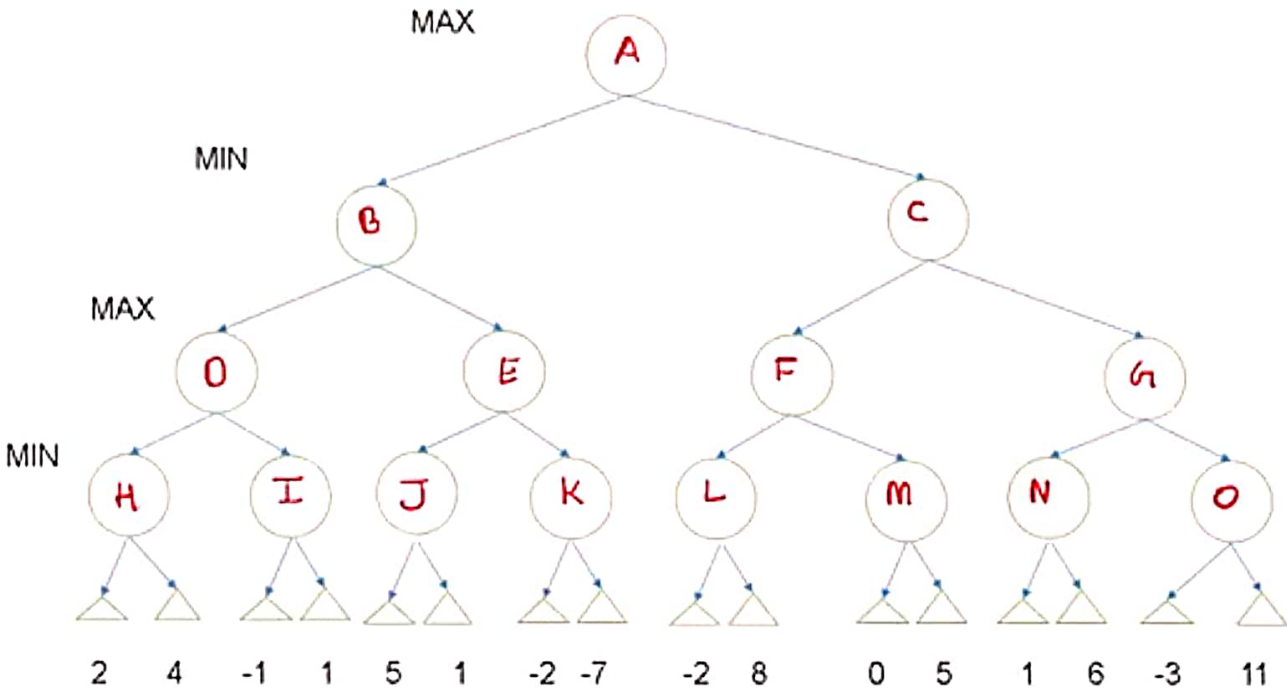
Min-Max Algorithm

- **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 $O(bm)$.

Alpha-Beta Pruning

- Alpha-beta pruning is an advance version of MINIMAX algorithm.
- The drawback of minimax strategy is that it explores each node in the tree deeply to provide the best path among all the paths. This increases its time complexity.
- But as we know, the performance measure is the first consideration for any optimal algorithm.
- Therefore, alpha-beta pruning reduces this drawback of minimax strategy by less exploring the nodes of the search tree.

Alpha-Beta Pruning.



Stochastic Games

- Many games are **unpredictable** in nature, such as those involving **dice throw**. These games are called as **Stochastic Games**. The outcome of the game depends on skills as well as luck.
- In the Stochastic Games, the winner of the game is not only decided by the skill but also by luck.
- Examples are
 - ❖ Gambling game
 - ❖ Golf ball game
 - ❖ Backgammon etc...

Stochastic Search Algorithms(Methods)

- Stochastic search algorithms are **designed for problems** with inherent random noise or deterministic problems solved by injected randomness.
- Desired properties of search methods are
 - ❖ High probability of finding near-optimal solutions (Effectiveness)
 - ❖ Short processing time (Efficiency)
- They are usually **conflicting**; a compromise is offered by stochastic techniques where certain steps are based on random choice.
- Many stochastic search techniques are inspired by processes found in nature. (Like Temperature)

Fully observable / Partially Observable

Fully observable

- An agent can always see the entire state of an environment.
- Example: Chess



Partially Observable

- An agent can never see the entire state of an environment.
- Example: Card game



What is an Agent?

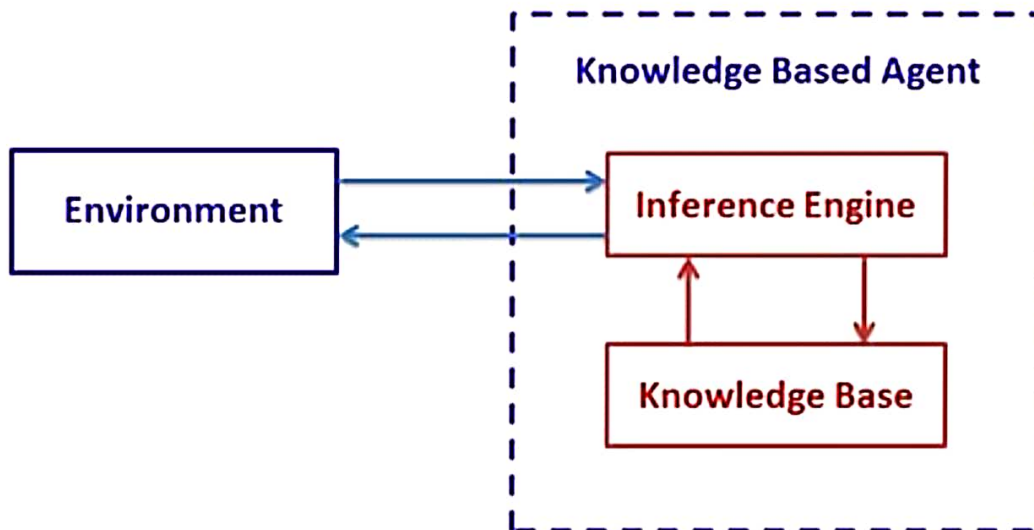
An agent can be anything that perceives environment through sensors and acts upon that environment through actuators.



Knowledge-Based Agent

- An intelligent agent needs **knowledge** about the real world for taking decisions and **reasoning** to act efficiently.
- Knowledge-based agents are those agents who have the capability of
 1. maintaining an internal state of knowledge,
 2. reason over that knowledge,
 3. update their knowledge after observations
 4. take actions.
- These agents can represent the world with some formal representation and act intelligently.

How it works?



Knowledge base




- A central component of a knowledge-based agent
- It is also known as KB.
- It is a collection of sentences
- These sentences are expressed in a language which is called a knowledge representation language.
- The Knowledge-base of KBA stores fact about the world.

Inference Engine

- Inference means deriving new sentences from old.
- A sentence is a proposition about the world.
- Inference engine allows us to add a new sentence to the knowledge base.
- Inference engine applies logical rules to the KB to deduce new information.
- Inference engine generates new facts so that an agent can update the KB.
- An inference engine works mainly in two rules which are given as:
 1. Forward Chaining and 2. Backward Chaining

What is the Wumpus World ?

- The wumpus world is a cave consisting of rooms connected by passageways.
- Lurking somewhere in the cave is the terrible wumpus, a beast that eats anyone who enters its room.
- The wumpus can be shot by an agent, but the agent has only one arrow.
- Some rooms contain bottomless pits that will trap anyone who wanders into these rooms (except for the wumpus, which is too big to fall in).
- The only mitigating feature of this bleak environment is the possibility of finding a heap of gold.

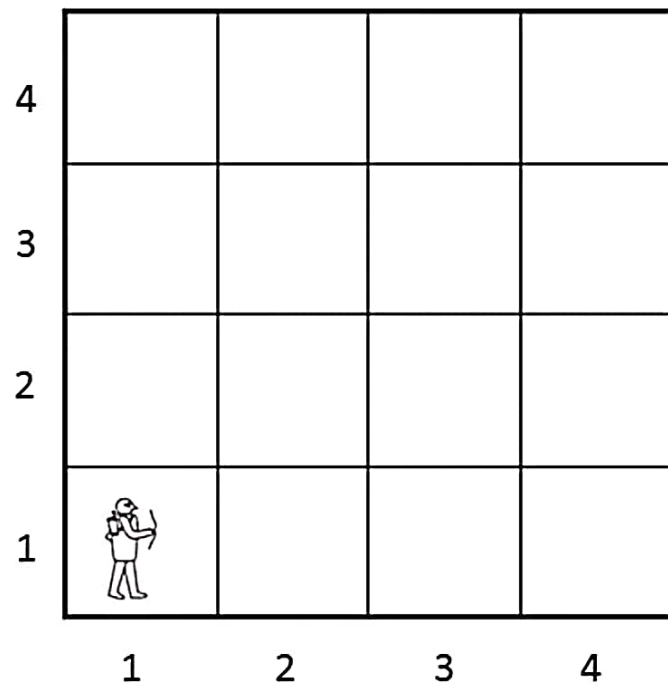
			PIT
		PIT	
		PIT	

Performance Measure

- +1000 for climbing out of the cave with the gold.
- -1000 for falling into a pit or being eaten by the wumpus.
- -1 for each action taken.
- -10 for using up the arrow.
- The game ends either when the agent dies or when the agent climbs out of the cave.

Environment

- A 4×4 grid of rooms.
- The agent always starts in the square labeled [1,1], facing to the right.
- The locations of the gold and the wumpus are chosen randomly, with a uniform distribution, from the squares other than the start square.
- In addition, each square other than the start can be a pit, with probability 0.2.






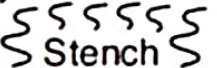
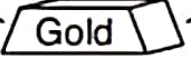










Actuators

- Agent can perform following actions:
 1. *Forward* : The agent can move *Forward*
 2. *TurnLeft* and *TurnRight* : The agent can *TurnLeft* by 90° , or *TurnRight* by 90° .
 3. *Grab* : The action *Grab* can be used to pick up the gold if it is in the same square as the agent.
 4. *Shoot* : The action *Shoot* can be used to fire an arrow in a straight line in the direction the agent is facing. The arrow continues until it either hits (and hence kills) the wumpus or hits a wall.

Sensors

- The agent has five sensors, each of which gives a single bit of information:
 1. In the square containing the wumpus and in the directly (not diagonally) adjacent squares, the agent will perceive a *Stench*.
 2. In the squares directly adjacent to a pit, the agent will perceive a *Breeze*.
 3. In the square where the gold is, the agent will perceive a *Glitter*.
 4. When an agent walks into a wall, it will perceive a *Bump*.
 5. When the wumpus is killed, it emits a woeful *Scream* that can be perceived anywhere in the cave.

 Stench		 Breeze	 PIT
	 Breeze  Stench  Gold	 PIT	 Breeze
 Stench		 Breeze	
 START	 Breeze	 PIT	 Breeze

Propositional Logic

- A very simple logic, here the statements are made by propositions.
- The propositional logic is also called as **Boolean Logic**.
- The sentence / statement is declarative, which is either **true or false**, but can not be both.
- Questions, opinion, and comma(,) are not allowed in propositional logic.
- E.g.
 - Students are studying in college (**True Proposition**)
 - $5 + 3 = 8$ (**True Proposition**)
 - `print("Hello World")` (**True Proposition**)
 - $4 + 2 = 5$ (**False Proposition**)
 - What is your name? (**not accepted / syntax error**)
 - Some students are intelligent (**false proposition because this declares both true / false**)

Propositional Logic - Syntax

- **Syntax** - defines the allowable sentences
- **The atomic sentences**- the individual syntactic elements- consist of a single proposition symbol.
- Each such symbol stands for a proposition that can be true or false.
- We will use uppercase names for symbols: P, Q, R, and so on.
- For example,
 - $W_{1,3}$ stand for the proposition that the **Wumpus** is in [1,3].

Propositional Logic - Syntax...

- **Complex sentences** : constructed from simpler sentences using logical connectives (logical operators)
- \neg (not) : called the **negation**, A literal is either an atomic sentence (a positive literal) or a negated atomic sentence (a negative literal).
- \wedge (and) : A sentence whose main connective is called a **conjunction**; its parts are the **conjuncts**.
- \vee (or) : a **disjunction** of the **disjuncts**
- \Rightarrow (implies) : is called an **implication** (or **conditional**).
- \Leftrightarrow (if and only if) : The sentence is a **biconditional**.

A Formal Grammar of Propositional Logic

- A BNF (Backus-Naur Form) grammar of sentences in propositional logic

$Sentence \rightarrow AtomicSentence \mid ComplexSentence$

$AtomicSentence \rightarrow \mathbf{True} \mid \mathbf{False} \mid \text{Symbol}$

$Symbol \rightarrow P \mid Q \mid R \mid \dots$

$ComplexSentence \rightarrow \neg Sentence$

$\mid (Sentence \wedge Sentence)$

$\mid (Sentence \vee Sentence)$

$\mid (Sentence \Rightarrow Sentence)$

$\mid (Sentence \Leftrightarrow Sentence)$

Propositional Logic - Semantics

- The semantics defines the rules for determining the truth of a sentence with respect to a particular model.
- In propositional logic, a model simply fixes the truth value - true or false - for every proposition symbol.
- $m_1 = (P_{1,2} = \text{false}, P_{2,2} = \text{false}, P_{3,1} = \text{true})$.

Propositional Logic – Semantics...

- The semantics for propositional logic must specify how to compute the truth value of any sentence, given a model.
- This is done **recursively**.
- All sentences are constructed from **atomic sentences** and **the five connectives**.
- Specify how to compute the **truth of atomic sentences** and how to compute the truth of sentences formed with each of the **five connectives**.
- Atomic sentences are easy.
- **Complex sentences** : For any sentence s and any model m , the sentence $\neg s$ is true in m if and only if s is false in m

Truth Tables for the Five Logical Connectives

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
<i>False</i>	<i>False</i>	<i>True</i>	<i>False</i>	<i>False</i>	<i>True</i>	<i>True</i>
<i>False</i>	<i>True</i>	<i>True</i>	<i>False</i>	<i>True</i>	<i>True</i>	<i>False</i>
<i>True</i>	<i>False</i>	<i>False</i>	<i>False</i>	<i>True</i>	<i>False</i>	<i>False</i>
<i>True</i>	<i>True</i>	<i>False</i>	<i>True</i>	<i>True</i>	<i>True</i>	<i>True</i>