



# Artificial Intelligence

---

Dr. Mubashir Ahmad (Ph.D.)

# Adversarial Search

- Adversarial search is a search, where we examine the problem which arises when we try to plan ahead of the world and other agents are planning against us.
- In previous topics, we have studied the search strategies which are only associated with a single agent that aims to find the solution which often expressed in the form of a sequence of actions.

# Adversarial Search

- But, there might be some situations where more than one agent is searching for the solution in the same search space, and this situation usually occurs in game playing.
- The environment with more than one agent is termed as **multi-agent environment**, in which each agent is an opponent of other agent and playing against each other. Each agent needs to consider the action of other agent and effect of that action on their performance.

# Adversarial Search

- So, Searches in which two or more players with conflicting goals are trying to explore the same search space for the solution, are called adversarial searches, often known as Games.
- **Deterministic games:** Deterministic games are those games which follow a strict pattern and set of rules for the games, and there is no randomness associated with them. Examples are chess, Checkers, Go, tic-tac-toe, etc.
- **Non-deterministic games:** Non-deterministic are those games which have various unpredictable events and has a factor of chance or luck. This factor of chance or luck is introduced by either dice or cards. These are random, and each action response is not fixed. Such games are also called as stochastic games. Example: Backgammon, Monopoly, Poker, etc.

# Zero-Sum Game

- Zero-sum games are adversarial search which involves pure competition.
- In Zero-sum game each agent's gain or loss of utility is exactly balanced by the losses or gains of utility of another agent.
- One player of the game try to maximize one single value, while other player tries to minimize it.
- Each move by one player in the game is called as ply.
- Chess and tic-tac-toe are examples of a Zero-sum game.

# Zero-Sum Game

- The Zero-sum game involved embedded thinking in which one agent or player is trying to figure out:
- What to do.
- How to decide the move
- Needs to think about his opponent as well
- The opponent also thinks what to do
- Each of the players is trying to find out the response of his opponent to their actions. This requires embedded thinking or backward reasoning to solve the game problems in AI.

# Formalization of the problem:

- **Initial state:** It specifies how the game is set up at the start.
- **Player(s):** It specifies which player has moved in the state space.
- **Action(s):** It returns the set of legal moves in state space.
- **Result(s, a):** It is the transition model, which specifies the result of moves in the state space.
- **Terminal-Test(s):** Terminal test is true if the game is over, else it is false at any case. The state where the game ends is called terminal states.
- **Utility(s, p):** A utility function gives the final numeric value for a game that ends in terminal states  $s$  for player  $p$ . It is also called payoff function. For Chess, the outcomes are a win, loss, or draw and its payoff values are  $+1$ ,  $0$ ,  $\frac{1}{2}$ . And for tic-tac-toe, utility values are  $+1$ ,  $-1$ , and  $0$ .

# Mini-Max Algorithm

- The minimax algorithm **helps find the best move, by working backwards from the end of the game**. At each step it assumes that player A is trying to maximize the chances of A winning, while on the next turn player B is trying to minimize the chances of A winning (i.e., to maximize B's own chances of winning).

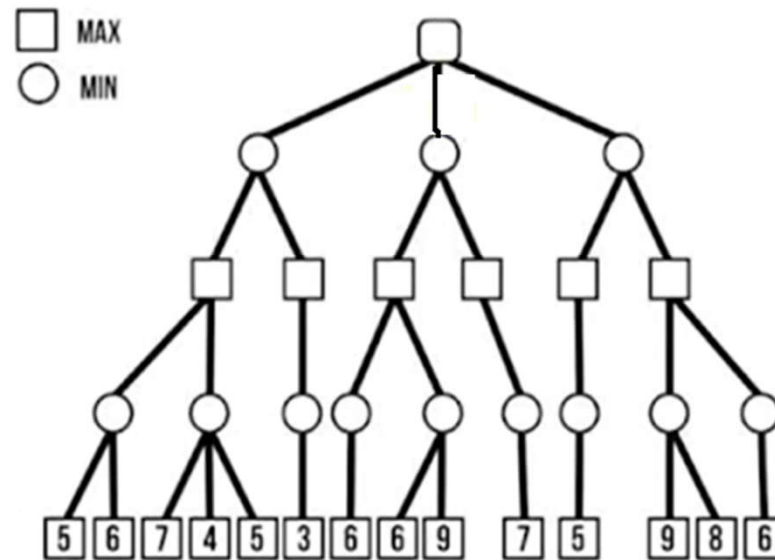


# Mini-Max Algorithm

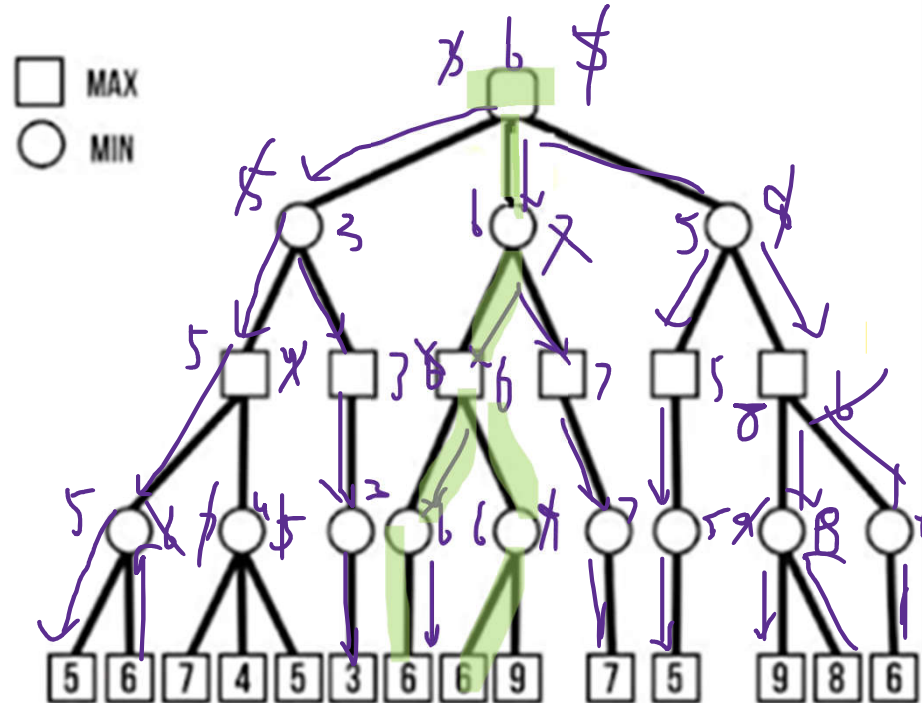
- Mini-max algorithm is a recursive or backtracking algorithm which is used in decision-making and game theory. It provides an optimal move for the player assuming that opponent is also playing optimally.
- Mini-Max algorithm uses recursion to search through the game-tree.
- Min-Max algorithm is mostly used for game playing in AI. Such as Chess, Checkers, tic-tac-toe, go, and various two player's game. This Algorithm computes the minimax decision for the current state.
- 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.
- The minimax algorithm proceeds all the way down to the terminal node of the tree, then backtrack the tree as the recursion.

# Mini-Max Algorithm Solved Example

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



# Mini-Max Algorithm Solved Example



# Mini-Max Algorithm

- 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  $O(b^m)$ .

# Alpha Beta Pruning Algorithm

- Alpha-beta pruning is **a search algorithm that seeks to decrease the number of nodes that are evaluated by the minimax algorithm in its search tree**. It is an adversarial search algorithm used commonly for machine playing of two-player combinatorial games (Tic-tac-toe, Chess, Connect 4, etc.).
- As we have seen in the minimax search algorithm that the number of game states it has to examine are exponential in depth of the tree. Since we cannot eliminate the exponent, but we can cut it to half. Hence there is a technique by which without checking each node of the game tree we can compute the correct minimax decision, and this technique is called **pruning**. This involves two threshold parameter Alpha and beta for future expansion, so it is called **alpha-beta pruning**. It is also called as **Alpha-Beta Algorithm**.

# Alpha Beta Pruning Algorithm

- Alpha-beta pruning can be applied at any depth of a tree, and sometimes it not only prune the tree leaves but also entire subtree.
- **Alpha:** The best (highest-value) choice we have found so far at any point along the path of Maximizer. The initial value of alpha is  $-\infty$ .
- **Beta:** The best (lowest-value) choice we have found so far at any point along the path of Minimizer. The initial value of beta is  $+\infty$ .

# Alpha Beta Pruning Algorithm

- Alpha-beta pruning is a modified version of the minimax algorithm. It is an optimization technique for the minimax algorithm.
- The Alpha-beta pruning to a standard minimax algorithm returns the same move as the standard algorithm does, but it removes all the nodes which are not really affecting the final decision but making algorithm slow. Hence by pruning these nodes, it makes the algorithm fast.

## **Key points about alpha-beta pruning:**

- The Max player will only update the value of alpha.
- The Min player will only update the value of beta.
- While backtracking the tree, the node values will be passed to upper nodes instead of values of alpha and beta.
- We will only pass the alpha, beta values to the child nodes.



# Alpha Beta Pruning Algorithm Solved Example

