**Common Terms in Game Theory:**

1. *Game:* Interaction b/w players where payoff depends on their decisions & others.
2. *Players:* Interdependent agents involved in the game.
3. *Actions:* Behaviors that players can choose from.
4. *Payoff:* Value changes based on players' actions.
5. *Zero-Sum:* One player's gain equals another's loss.
6. *Non-Zero-Sum:* Net benefit or loss to the system.
7. *Simultaneous:* Players make decisions at the same time.
8. *Sequential:* Players alternate in making decisions.
9. *Non-Cooperative:* Competitive game among individual players.
10. *Cooperative:* Players can form alliances and respond cooperatively.
11. *Complete Information:* All participants have knowledge about the game.
12. *Incomplete Information:* Players lack information on certain aspects of the game.
13. *Imperfect Information:* Players are unaware of others' chosen actions.

**Adversarial Search:**

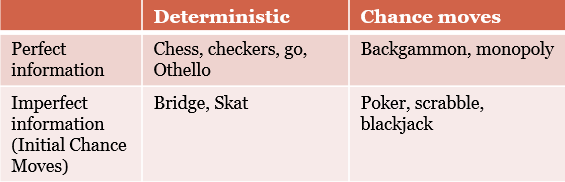
Planning in a world where other agents are planning against us. Often seen in board games where two agents alternate actions. Assumes zero-sum games of perfect information in fully observable environments. AI assumptions include alternating actions, utility values, and fully observable environments.

**Search versus Games:**

*Search:* No adversary, solution is a method for finding goals, and heuristic techniques can find optimal solutions.

*Games:* Involves adversaries, solution is a strategy, optimality depends on the opponent, time limits force approximate solutions.

**Types of Games:**

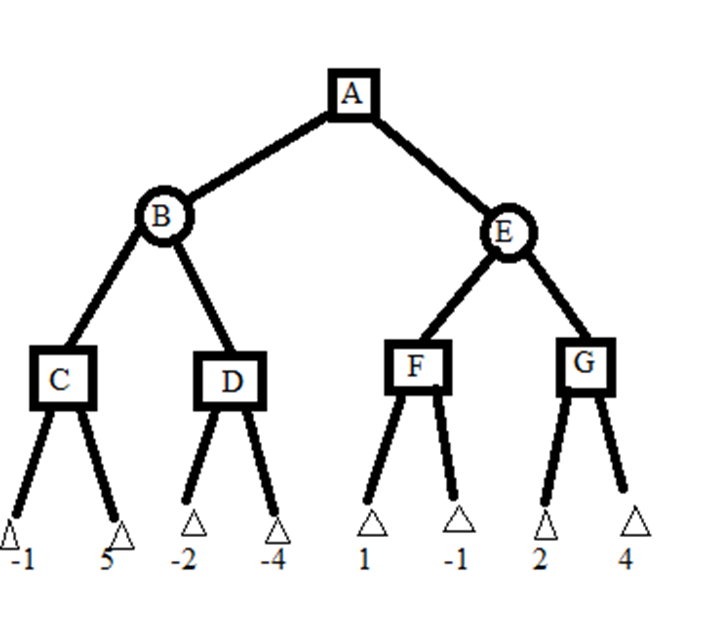
Every game with chance moves during the game has equivalent representation with initial chance moves only. While this is a deep result, it's more tractable to consider chance moves as the game progresses.

**Game Setup:** Games are treated as search problems.

1. Initial state: Initial position, such as the board configuration in chess.
2. Successor function: Lists legal moves from any position.
3. Terminal test: Determines if the game is over.
4. Utility function: Provides a numerical value for terminal states, such as win (+1), lose (-1), and draw (0) in tic-tac-toe or chess.

**Basic Strategy:**

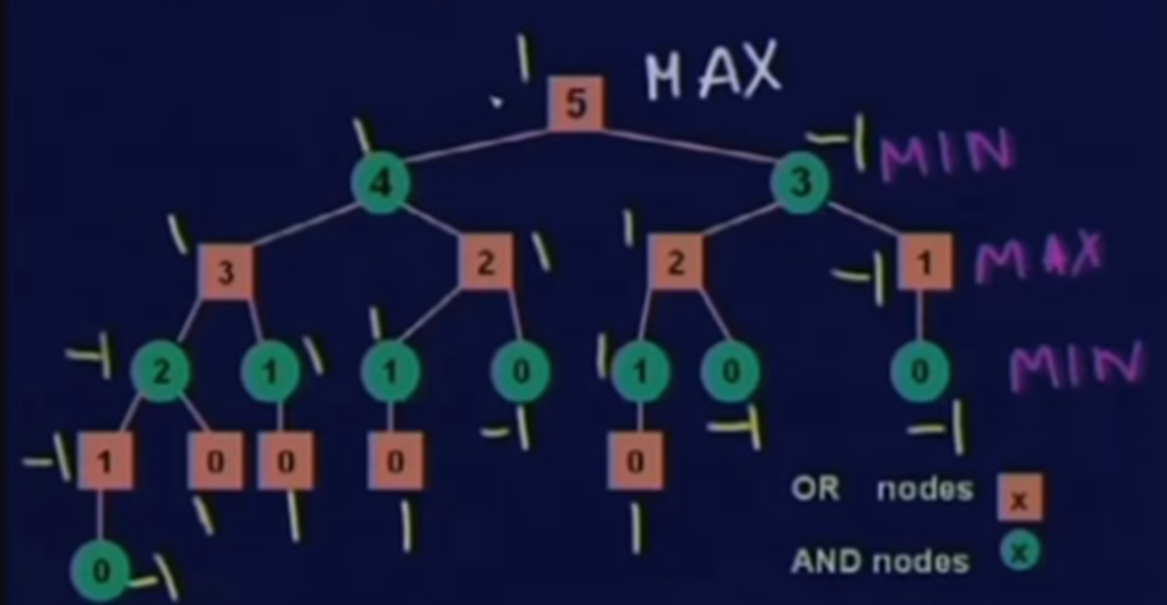
1. *Grow a Search Tree:* Only one player moves at each turn. Payoffs are assigned to final positions, known as utility. Values are propagated from final positions.
2. *Assume the opponent makes moves worst for us:* Pick the best moves on your turn.

**2-Player Games:**

Involves two players in a zero-sum, perfect information game. Players are MAX and MIN, taking turns to maximize and minimize a utility function. MAX player makes the first move and turns alternate until the game ends. Winner receives an award, & loser faces a penalty.

Game tree consists of nodes representing positions where players must move. Nodes are classified as MAX or MIN, depending on the player's turn. Game tree could potentially be infinite. Ply of a node represents the number of moves needed to reach that node from root.

**Brute Force Search:**

Start at the root node and generate the entire search tree until the leaf positions, assuming the tree is finite. Feasible for small games but not practical for large games due to the potential for an infinite tree. Provides a foundation for further discussion & strategies.

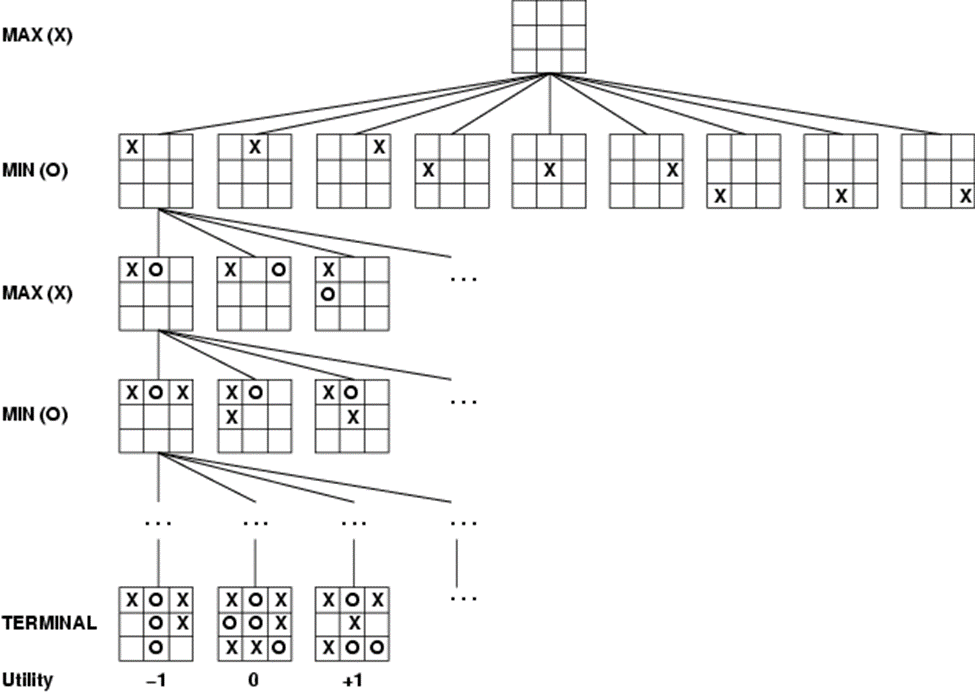
1. *5-Stone Nim*

Played with two players and a pile of stones.

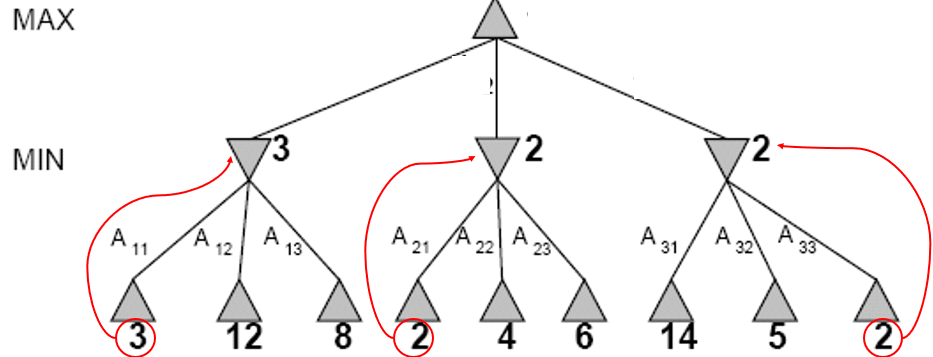
Each player removes 1 or 2 stones from the pile.

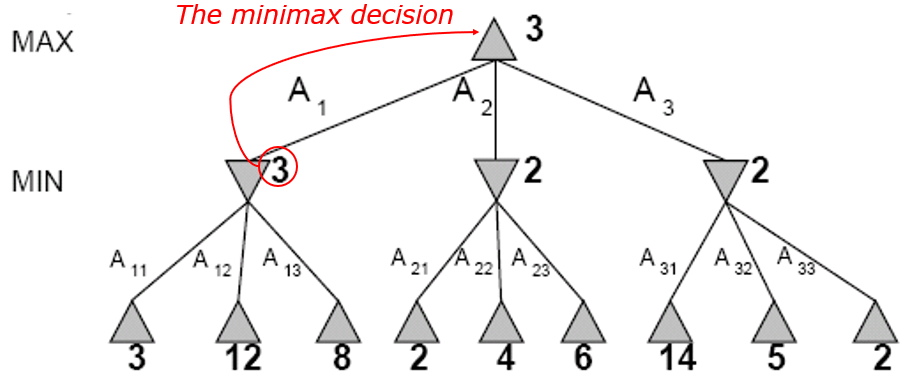
The player who removes the last stone wins the game.

1. *2-player, deterministic, turns:-*

**

1. *Two-Ply Game Tree:-*



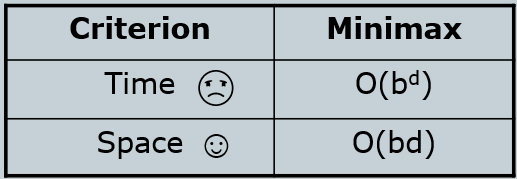


**Minimax Algorithm:** It involves a complete depth-first exploration of the game tree.

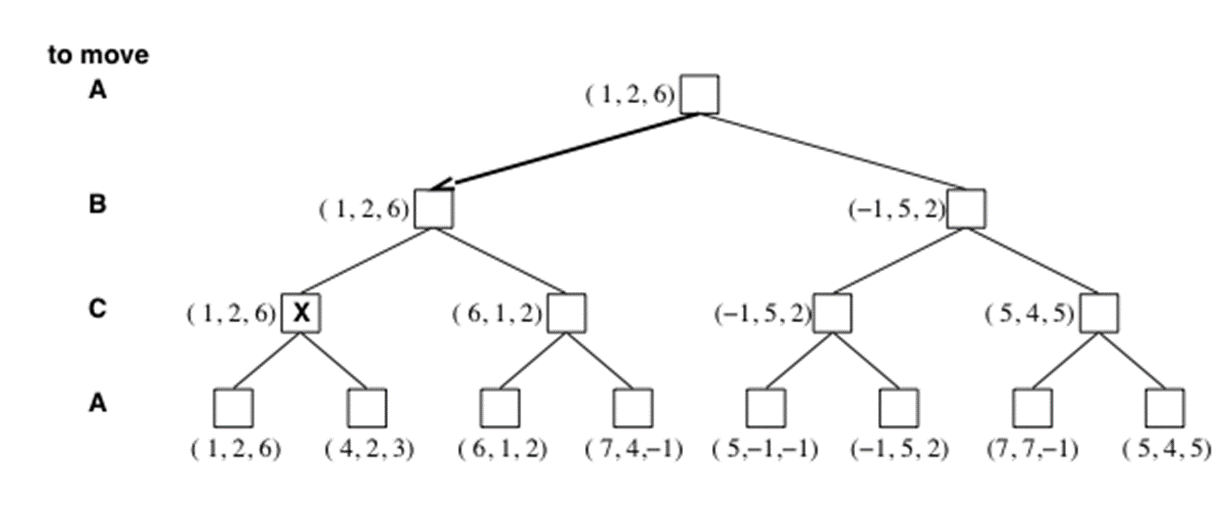
Maximum depth = d

b legal moves at each point

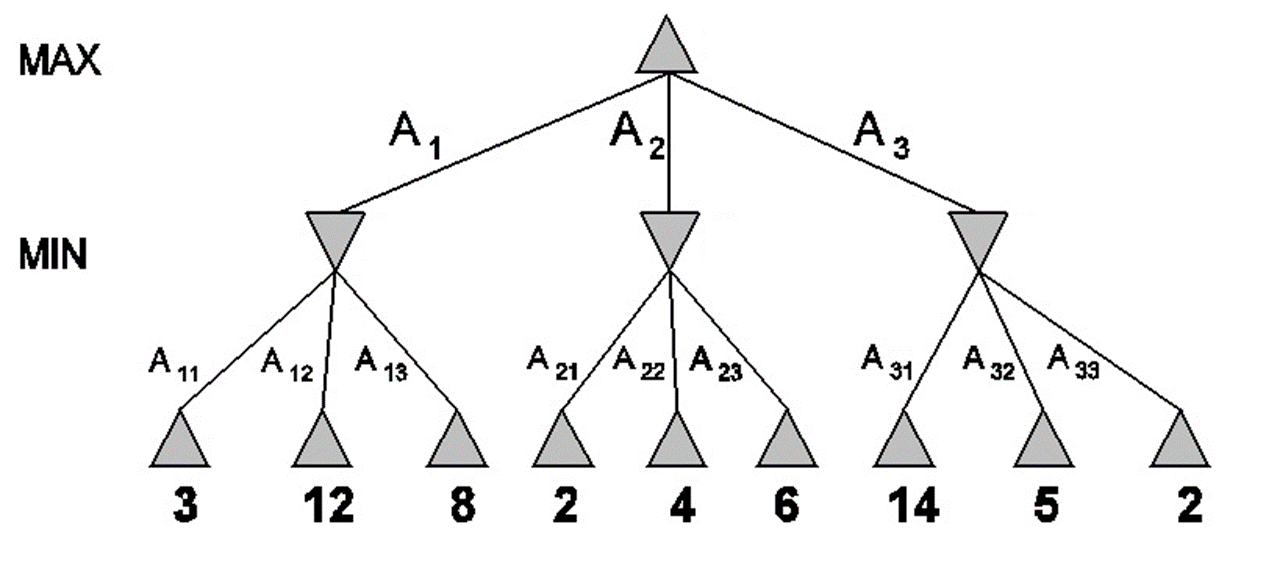
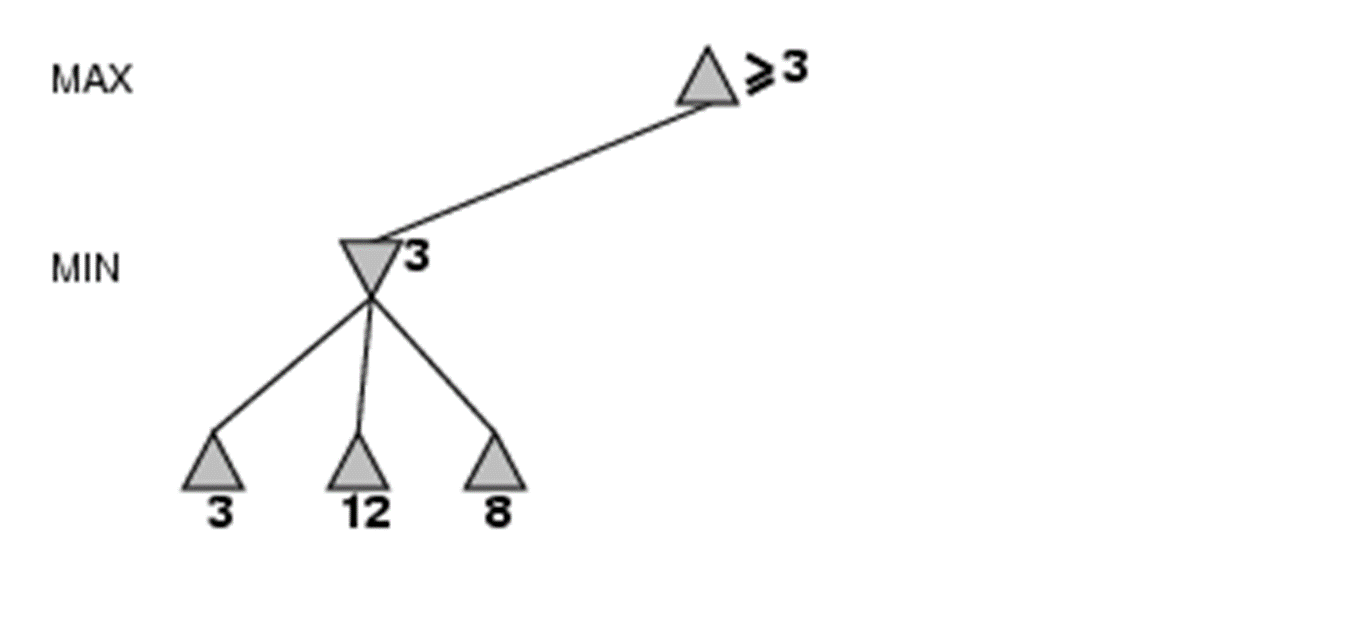
Example: In chess, d is approximately 100, and b is around 35.

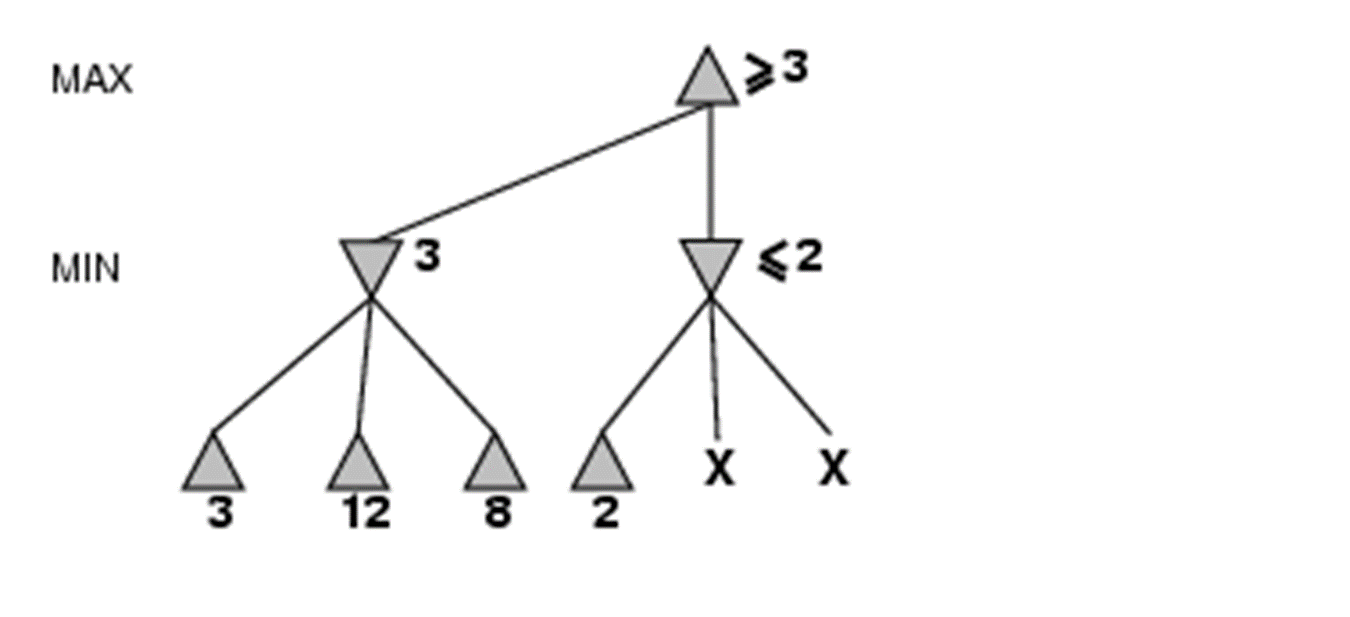
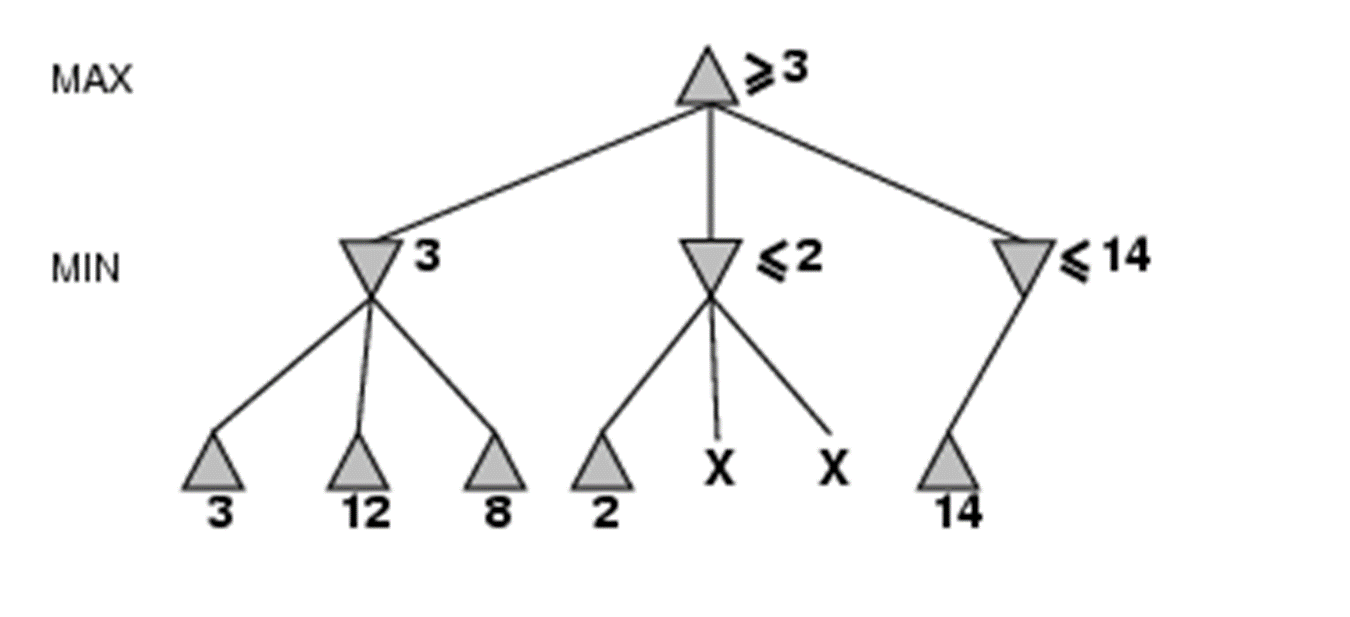
*Challenges:*

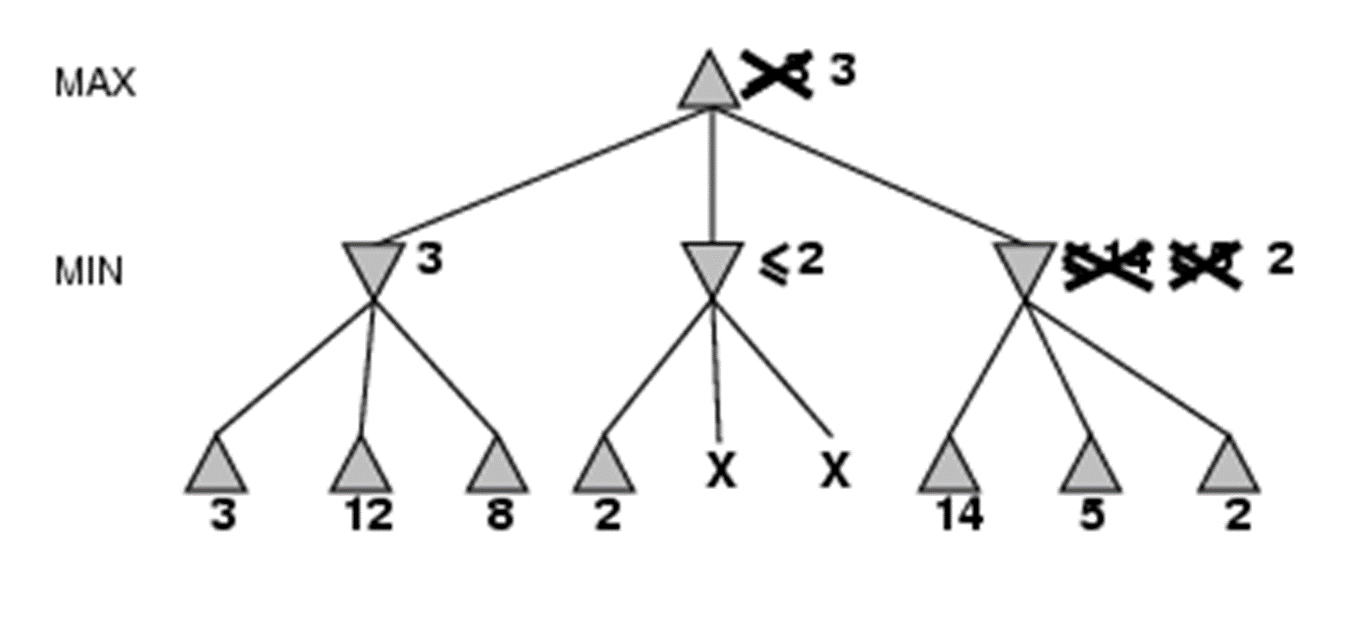
* The number of game states is exponential in the number of moves.
* Solution: Pruning - remove branches that do not influence the final decision.



**Pruning example:-**

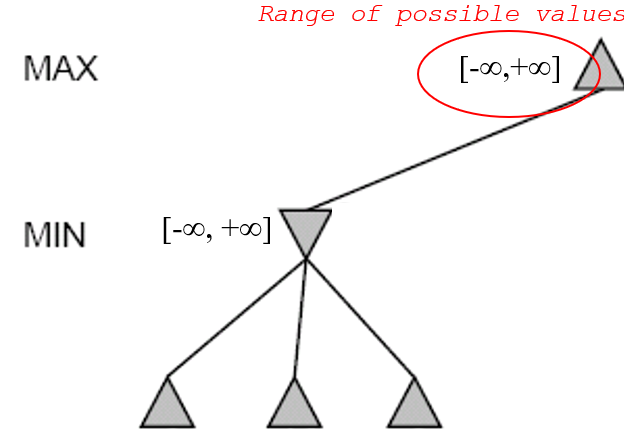
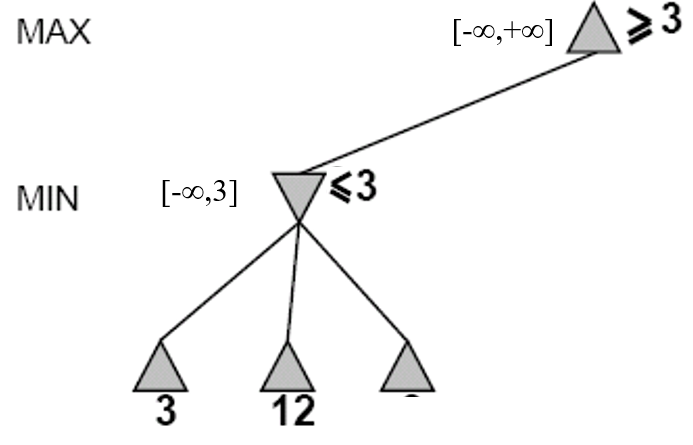
 

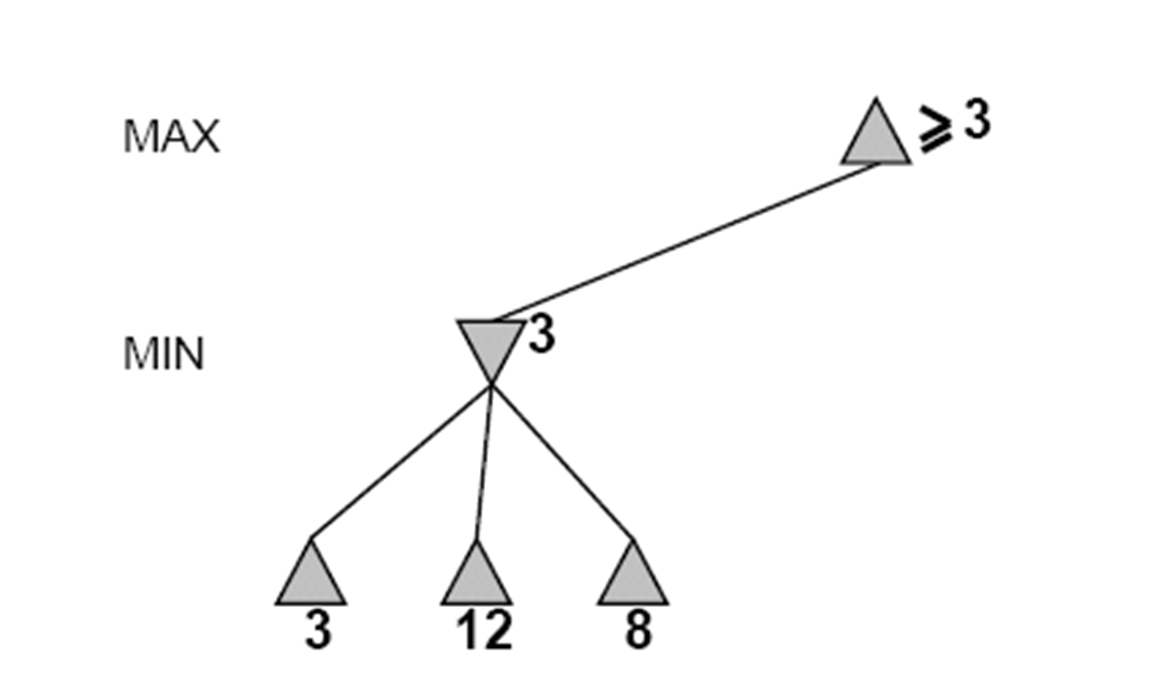
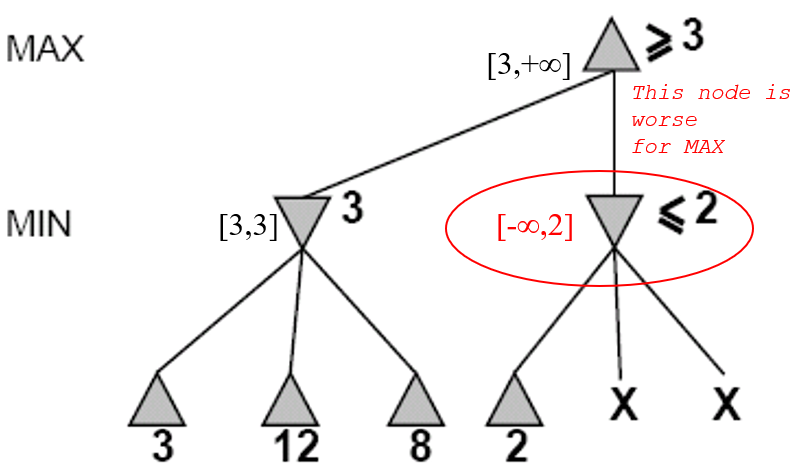


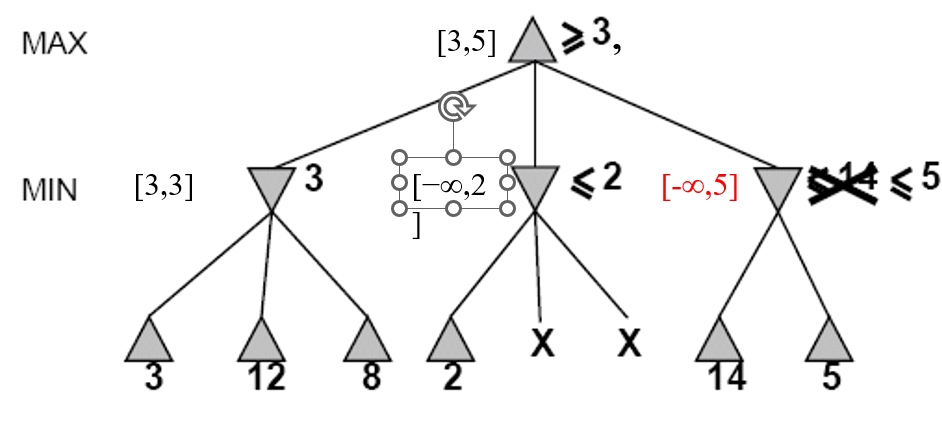
**The Alpha-beta Algorithm:**

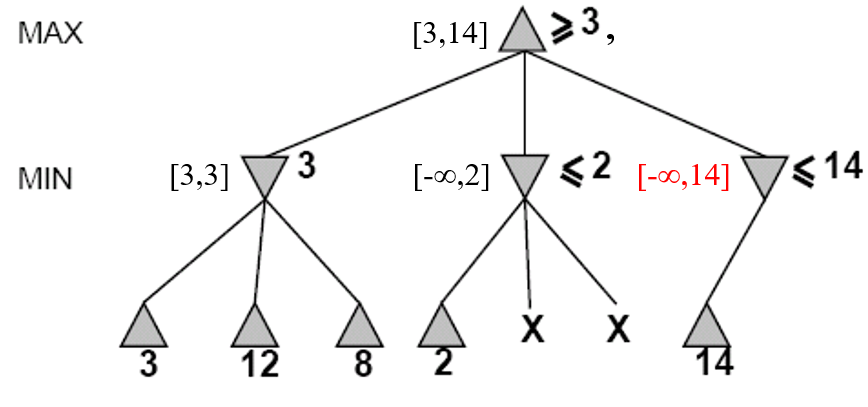
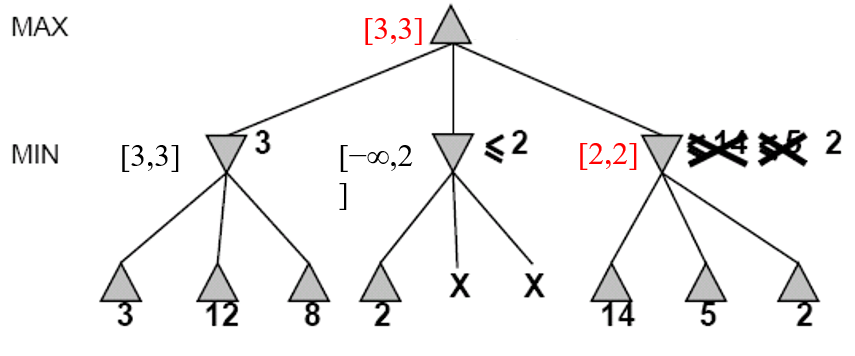
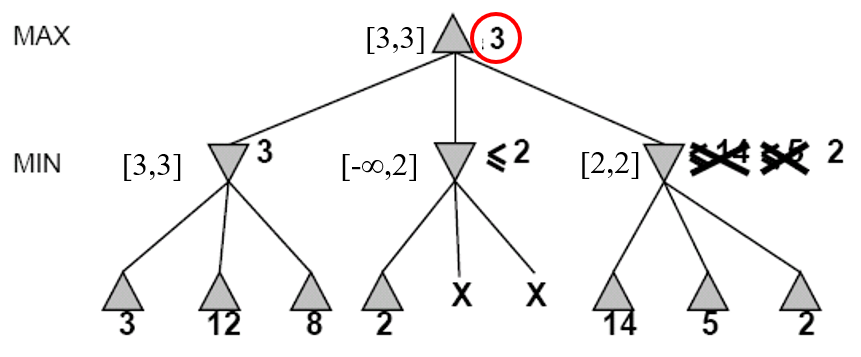
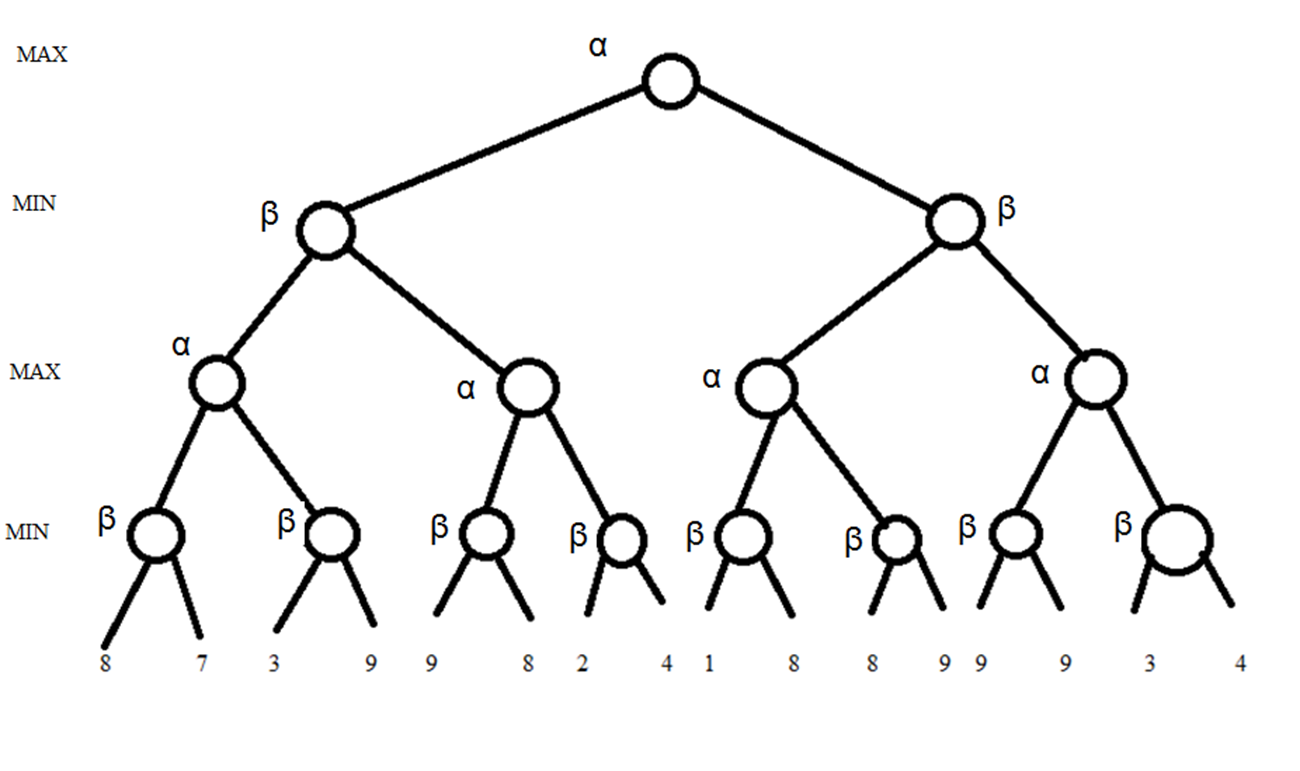
1. Conducts depth-first search, considering nodes along a single path.
2. Uses parameters α (alpha) and β (beta) to bound backed-up values along the path.
3. Prunes branches as soon as a value is known to be worse than the current α or β value for MAX or MIN, respectively.
4. Pruning doesn't affect the final result.
5. Its effectiveness depends on the order of examining states.
6. Good move ordering enhances pruning effectiveness.
7. With perfect ordering, time complexity is O(b^m/2) (b: branching factor, m: depth).
8. It doubles the depth of search compared to minimax in the same time.
9. Random order examining results in O(b^(3m/4)) nodes examined for moderate b.
10. Techniques like killer moves and transposition tables can improve performance and enable self-learning.

Same example as above:-









**Local Search Algorithms and Optimization Problems:**

1. *Hill Climbing:*

Iterative optimization algorithm. Starts with an arbitrary solution and iteratively moves to a neighboring solution with a better objective value.

It halts when it reaches a peak where no better solution can be found.

Simple and easy to implement, but can get stuck in local optima and lacks global perspective.

1. *Simulated Annealing:*

Inspired by the annealing process in metallurgy.

Allows for escaping local optima by accepting worse solutions with a certain probability.

The probability of accepting worse solutions decreases over time (annealing schedule).

Balances between exploration and exploitation.

Provides a good chance to find global optima but requires careful tuning of parameters.

1. *Genetic Algorithms (GAs):*

Inspired by the principles of natural selection and genetics.Operates on a population of potential solutions represented as individuals.

Utilizes genetic operators such as selection, crossover, and mutation to evolve the population.

Encourages diversity within the population to explore the search space effectively.

Suitable for a wide range of optimization problems but requires careful parameter tuning and can be computationally expensive.