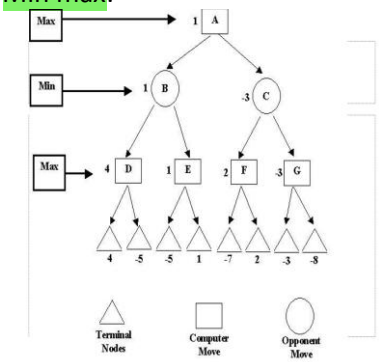
Games/Adversarial search

5 steps

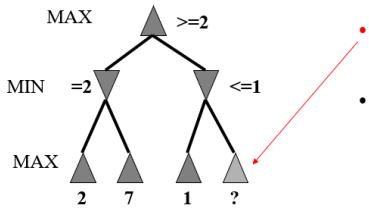
Pros:	Example:	Characteristics:	How to play:	Evaluation function:
When very large spaces, BFS is not used Solve hard problems with minimal initial structure Min max:  Backtracking algo – DFS Optimal Complete Time complexity: $O(b^m)$ Space complexity: O(bm) b-legal moves m-depth	- Chess - Checkers - Go - Bridge - Nim - Tic-tac-toe - Othello Agents 2 types: Competitive Cooperative	- 2 person game (multiagent environments) - Alternate moves (one after another) - Zero sum - Perfect information - No chance involved Game tree: a tree where the nodes are game states and the edges are moves	Consider all the legal moves you can make. Compute the new position resulting from each move. Evaluate each resulting position and determine which is best. Make that move. Wait for your opponent to move and repeat.	**f(n) >> 0**: position n good for me and bad for you **f(n) << 0**: position n bad for me and good for you **f(n) near 0**: neutral position **f(n) = +infinity**: win for me **f(n) = -infinity**: win for you

Alpha beta pruning: improve performance of minmax.

Gives same value of the root node as minmax algo, but needs less or equal computation.

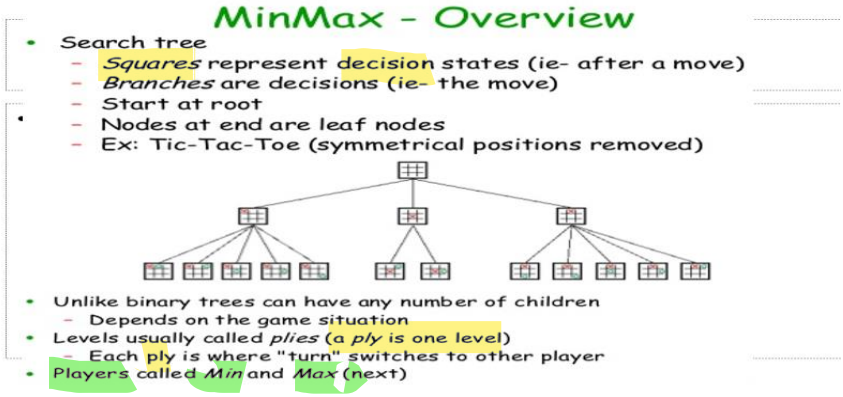
DFS algo follow, time complexity: $O(b^{m/2})$

- **Worst case:** no pruning, examining b^d leaf nodes, where each node has b children and a d-ply search is performed
- **Best case:** examine only $(2b)^{d/2}$ leaf nodes.
- **Best case** is when each player's best move is the first alternative generated



- We don't need to compute the value at this node.
- No matter what it is, it can't affect the value of the root node.

- At each **MAX** node n, **alpha(n)** = maximum value found so far, start at -infinity and only increase
- At each **MIN** node n, **beta(n)** = minimum value found so far, start at +infinity and only decrease.



Extra:

1) Zero sum: one's loss is other's gain.
2) Perfect Information: both players have access to complete information about the state of the game.
3) No chance: players have full control over their moves and the outcome of the game is solely determined by their strategic decisions and actions.
4) **Evaluation function / static evaluator**: evaluate the "goodness" of a game position.
5) Heuristic evaluation function: approximate the true utility of a state without doing a complete search. Ex: pruning
6) **Alpha cutoff:** stop searching below MIN node n if beta(n) <= alpha(i) for some MAX node ancestor i of n.
7) **Beta cutoff:** stop searching below MAX node n if beta(i) <= alpha(n) for some MIN node ancestor i of n.
8) Pruning: ignore portions of the search tree that make no difference to the final choice
9) Metareasoning: It involves the ability to monitor, evaluate, and control one's own cognitive activities to improve problem-solving, decision-making, and learning.