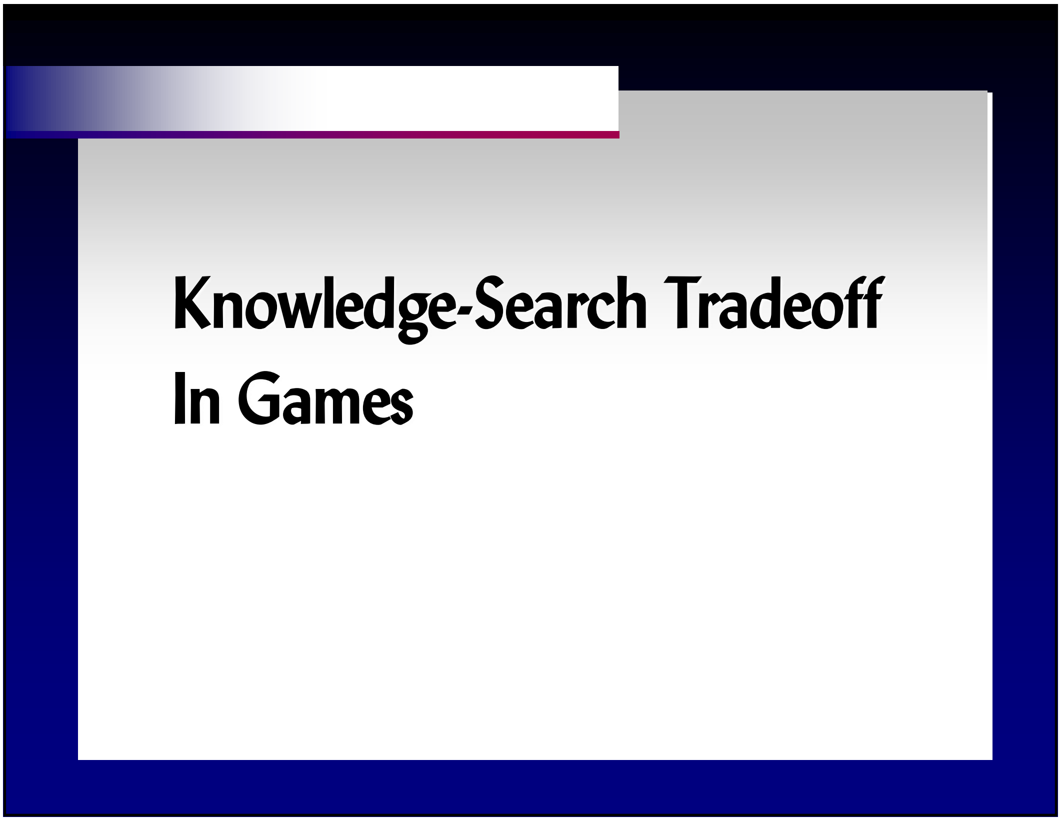
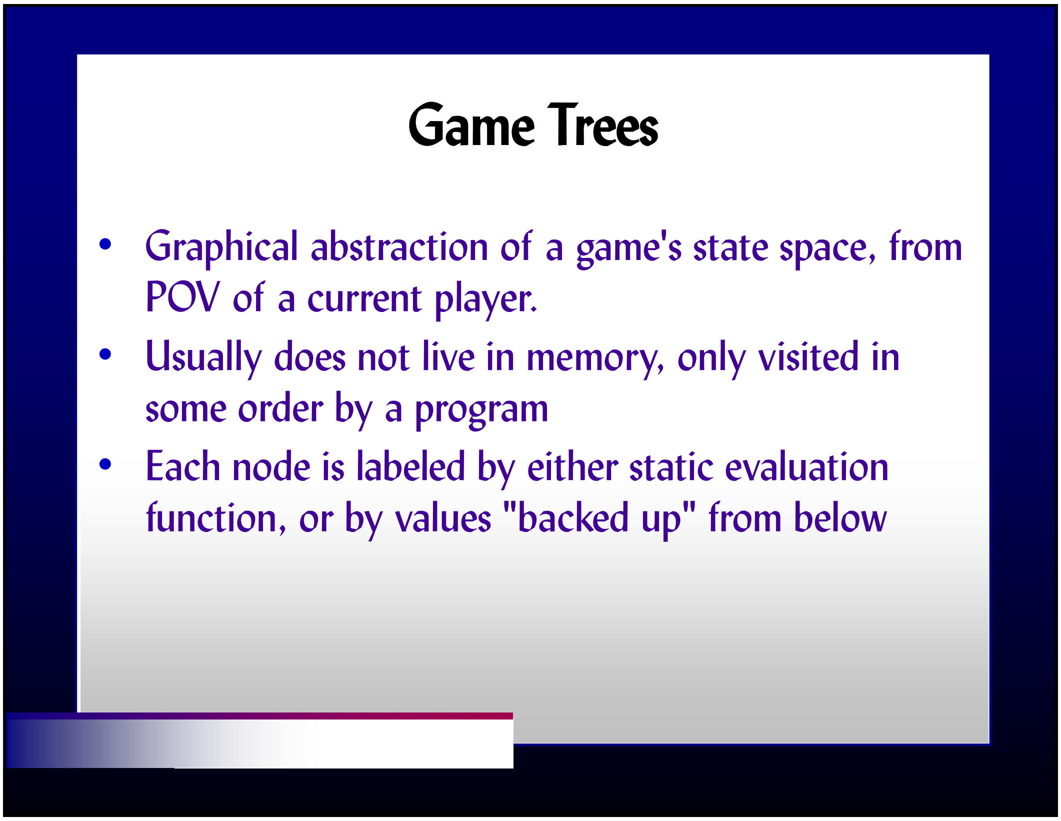
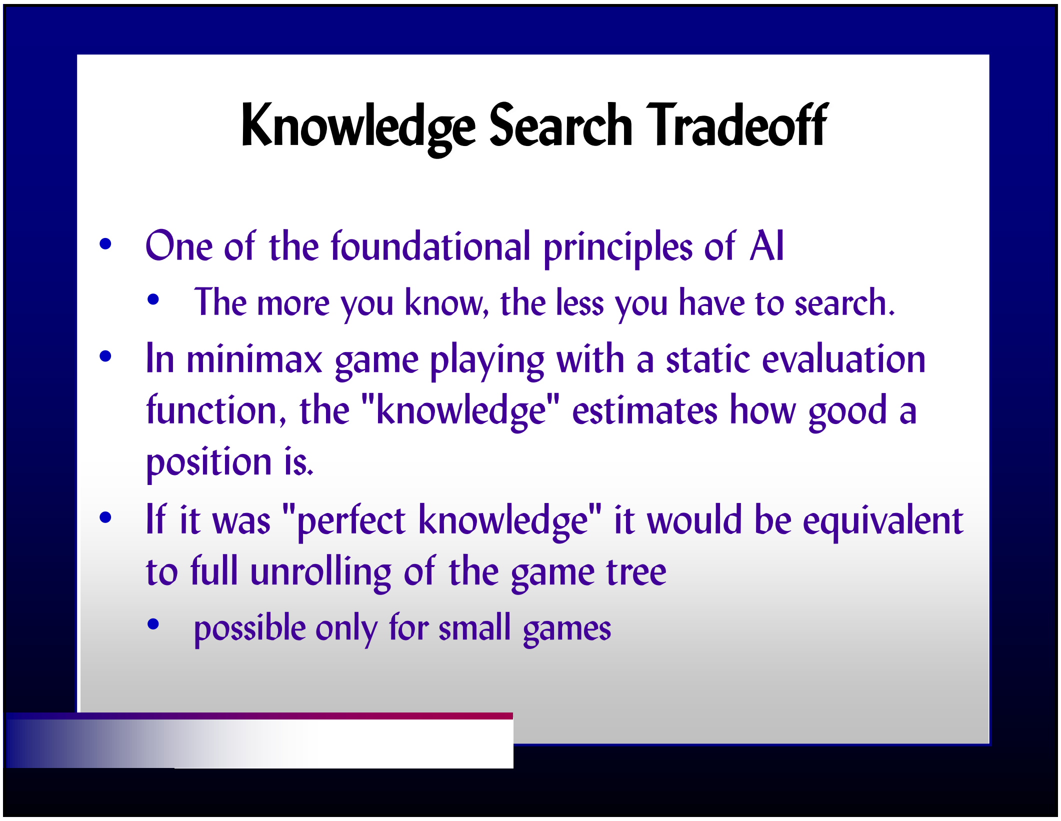


Knowledge-Search Tradeoff In Games



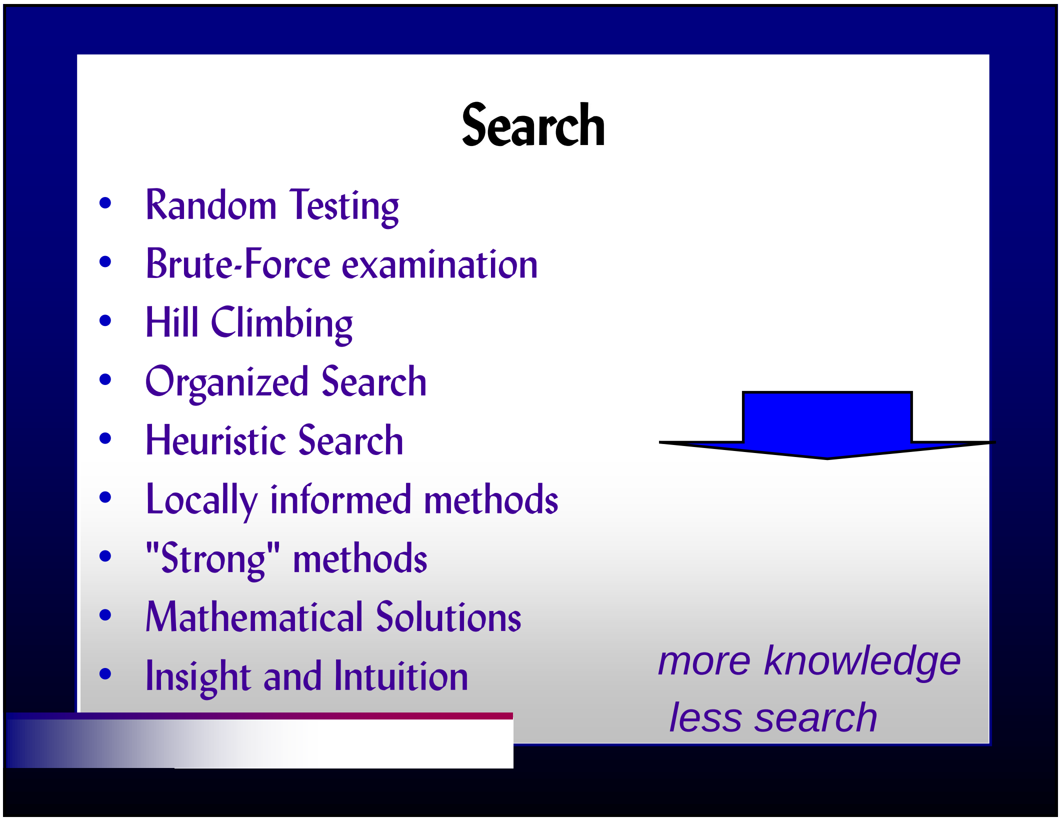
Game Trees

- Graphical abstraction of a game's state space, from POV of a current player.
- Usually does not live in memory, only visited in some order by a program
- Each node is labeled by either static evaluation function, or by values "backed up" from below



Knowledge Search Tradeoff

- One of the foundational principles of AI
  - The more you know, the less you have to search.
- In minimax game playing with a static evaluation function, the "knowledge" estimates how good a position is.
- If it was "perfect knowledge" it would be equivalent to full unrolling of the game tree
  - possible only for small games

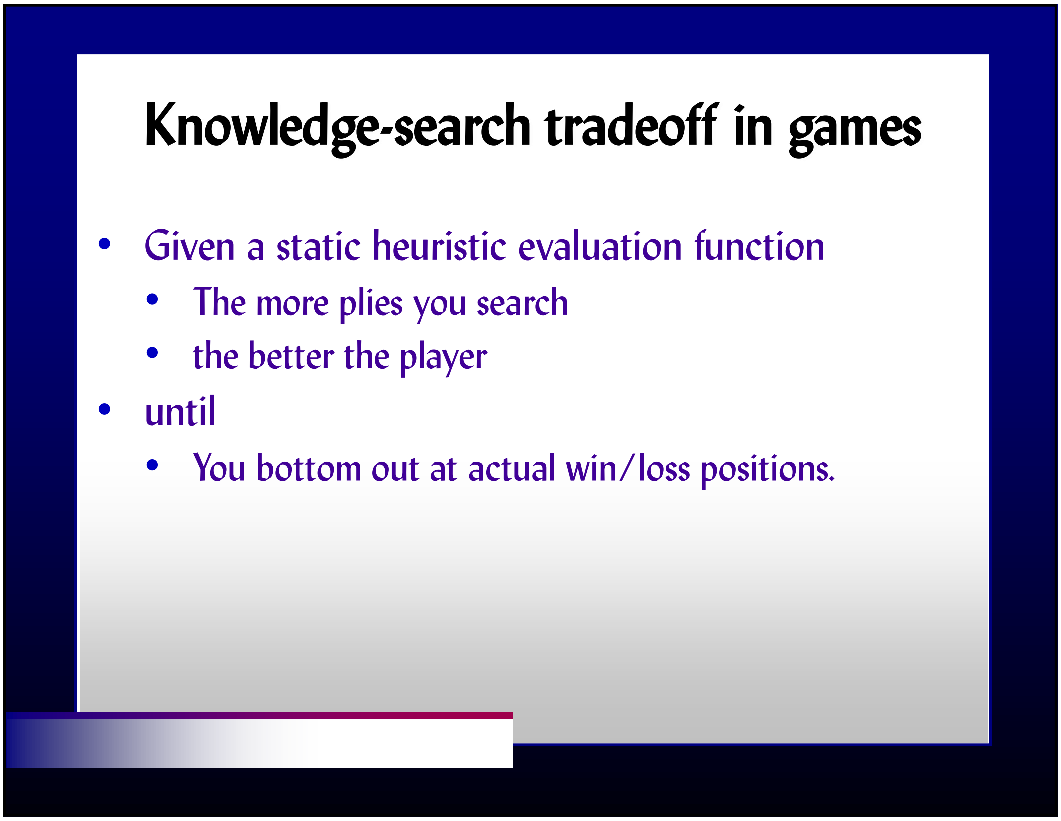


Search

- Random Testing
- Brute-Force examination
- Hill Climbing
- Organized Search
- Heuristic Search
- Locally informed methods
- "Strong" methods
- Mathematical Solutions
- Insight and Intuition

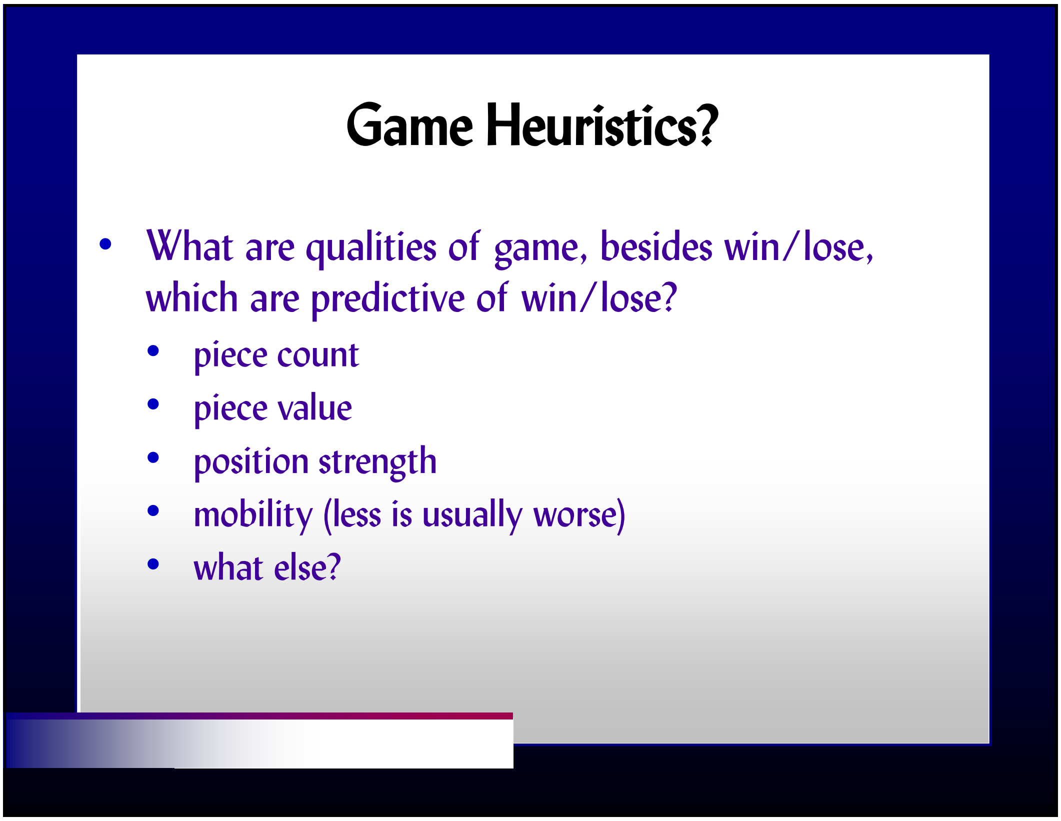
more knowledge

less search



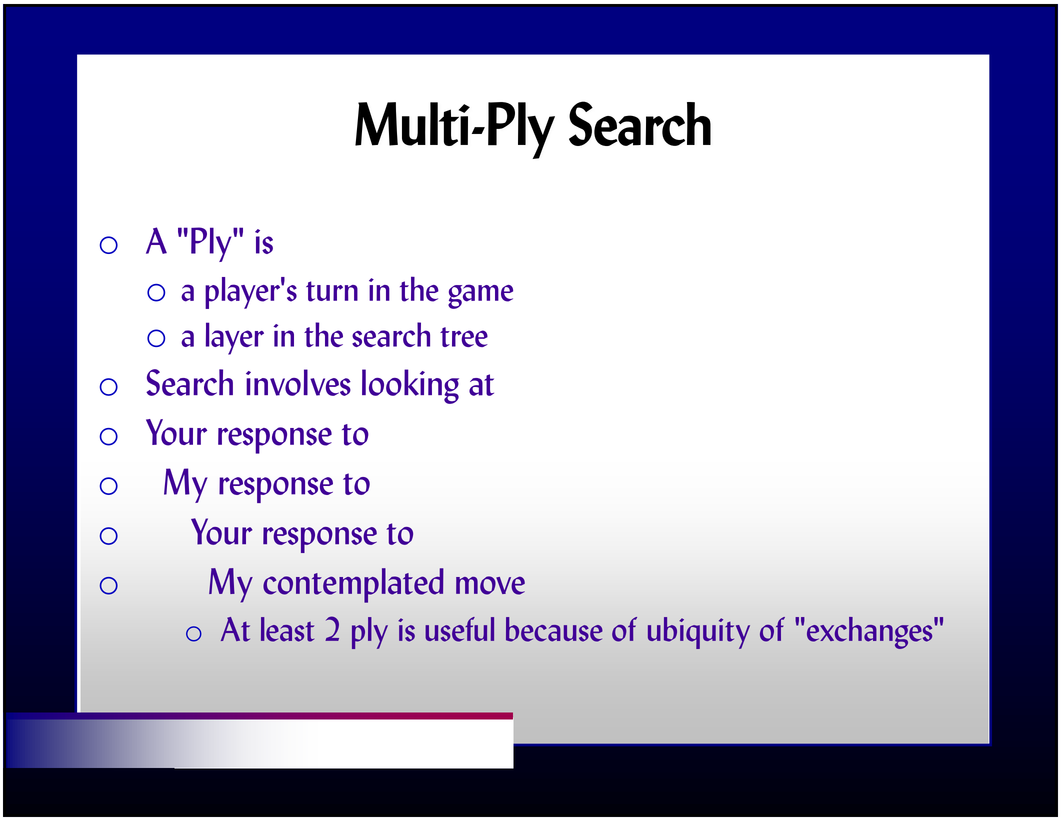
Knowledge-search tradeoff in games

- Given a static heuristic evaluation function
  - The more plies you search
  - the better the player
- until
  - You bottom out at actual win/loss positions.



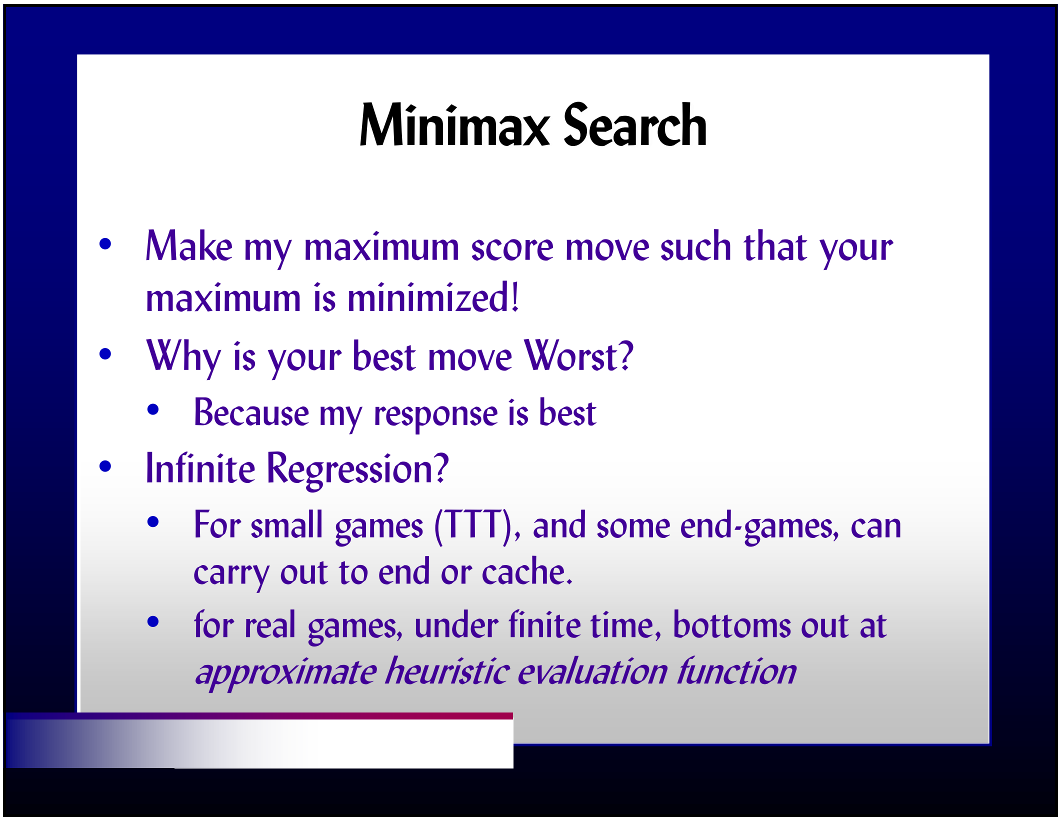
Game Heuristics?

- What are qualities of game, besides win/lose, which are predictive of win/lose?
  - piece count
  - piece value
  - position strength
  - mobility (less is usually worse)
  - what else?



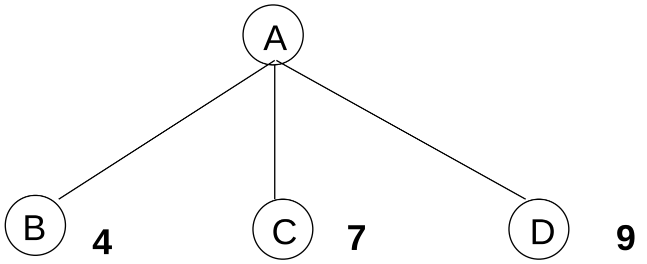
Multi-Ply Search

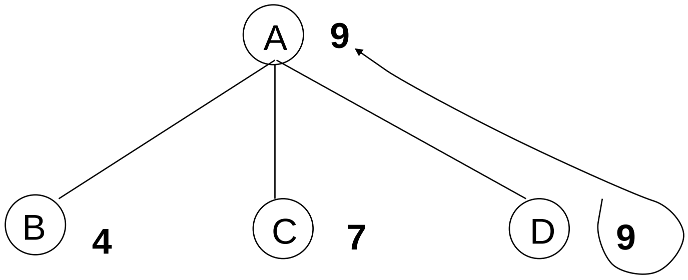
- A "Ply" is
  - a player's turn in the game
  - a layer in the search tree
- Search involves looking at
- Your response to
- My response to
- Your response to
- My contemplated move
  - At least 2 ply is useful because of ubiquity of "exchanges"



Minimax Search

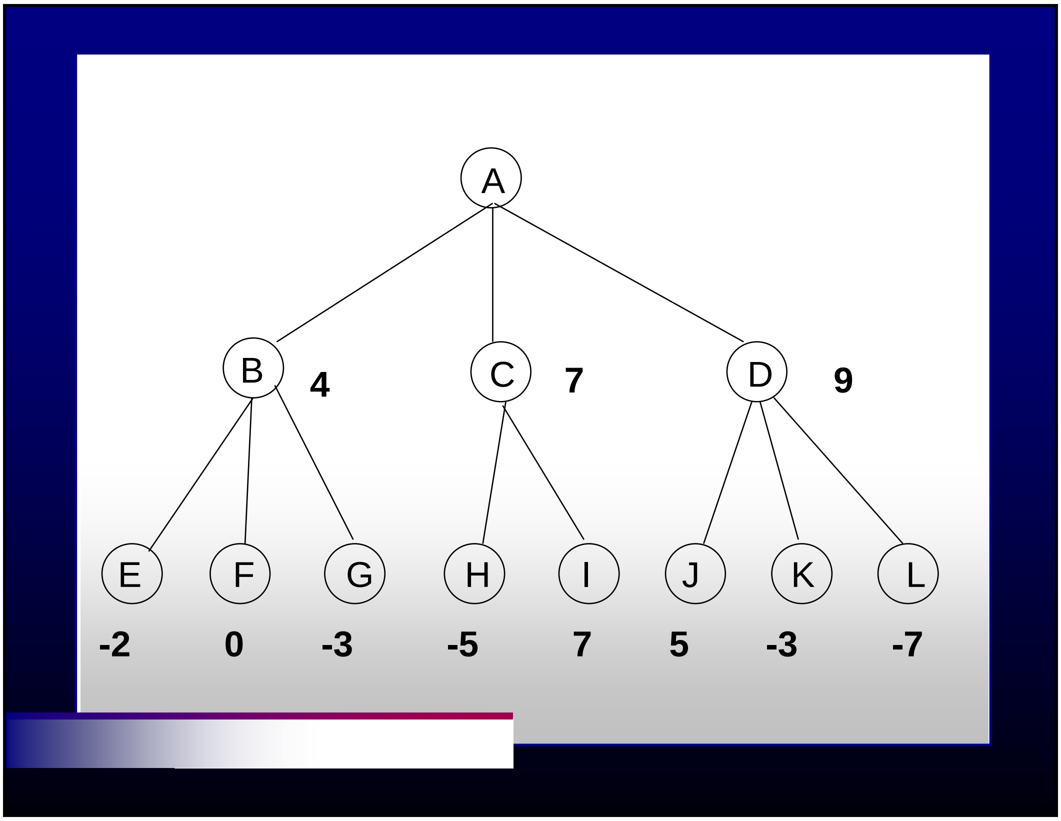
- Make my maximum score move such that your maximum is minimized!
- Why is your best move Worst?
  - Because my response is best
- Infinite Regression?
  - For small games (TTT), and some end-games, can carry out to end or cache.
  - for real games, under finite time, bottoms out at *approximate heuristic evaluation function*

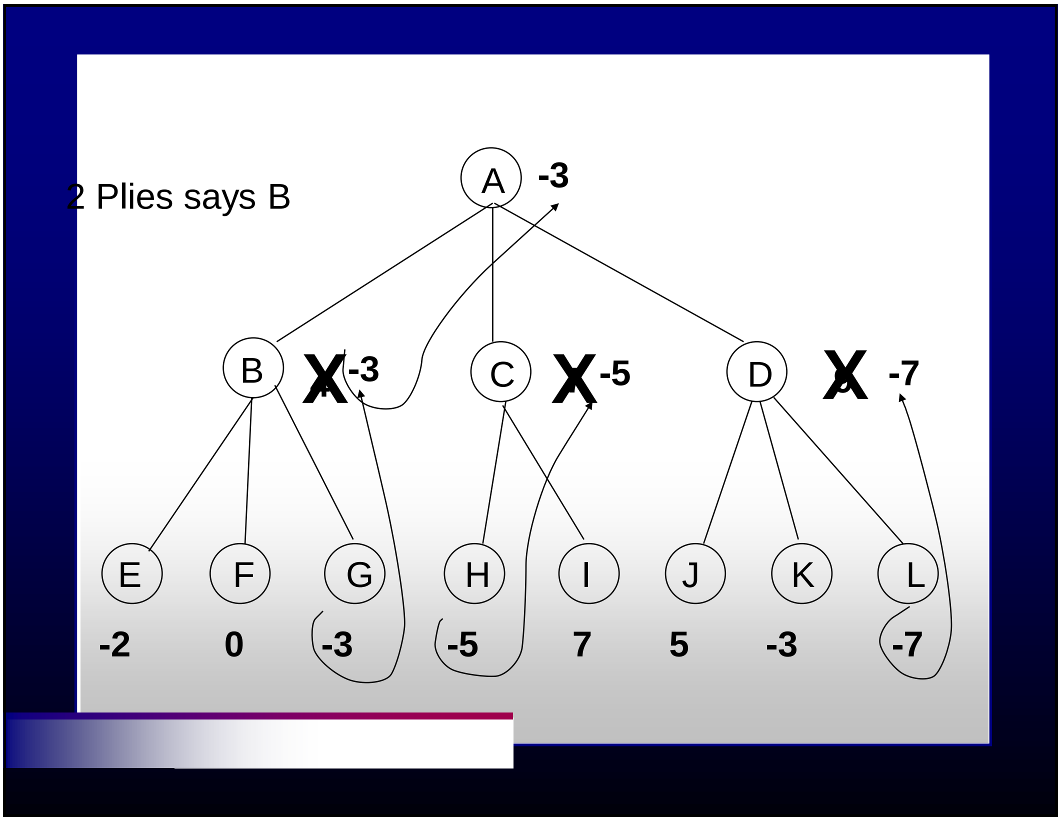


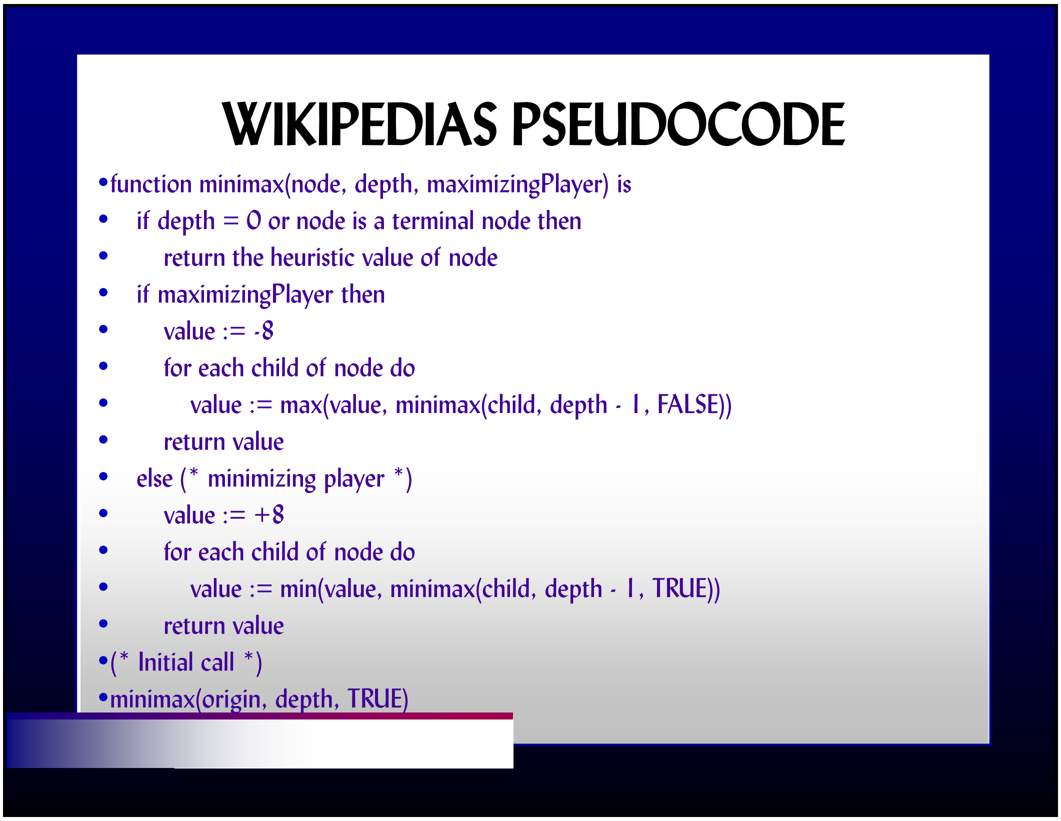


What is my Best Move?

D maximizes the score!

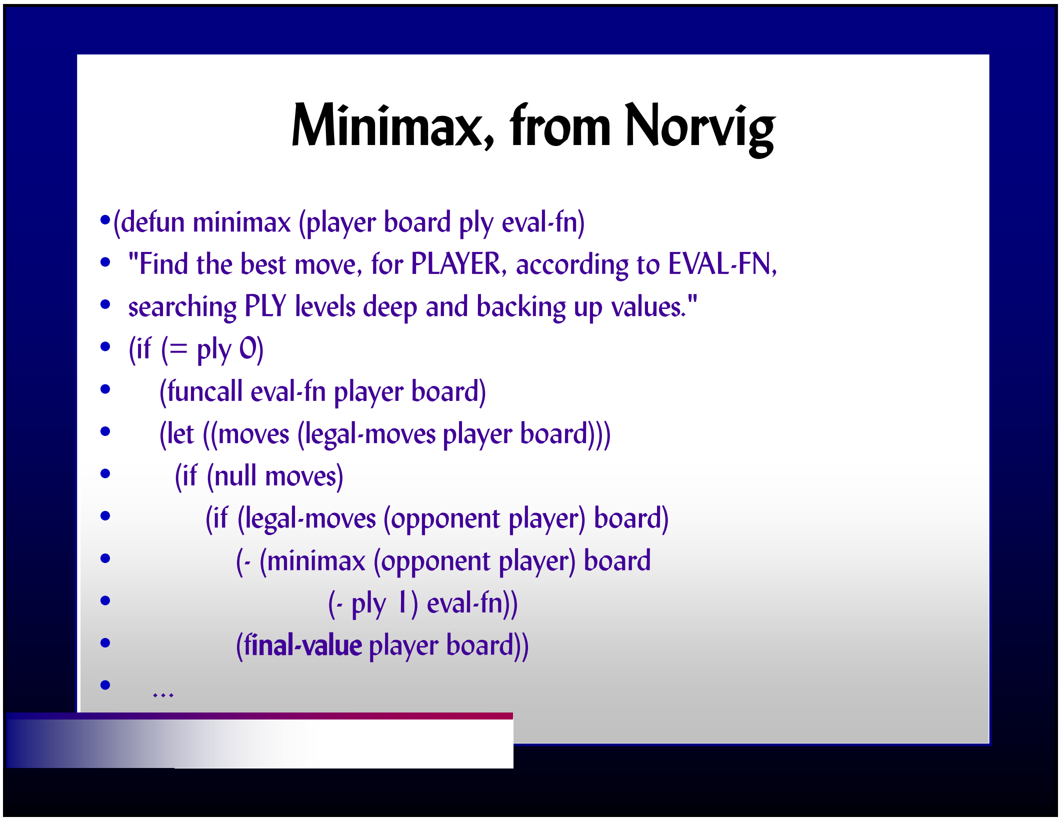



2 Plies says B



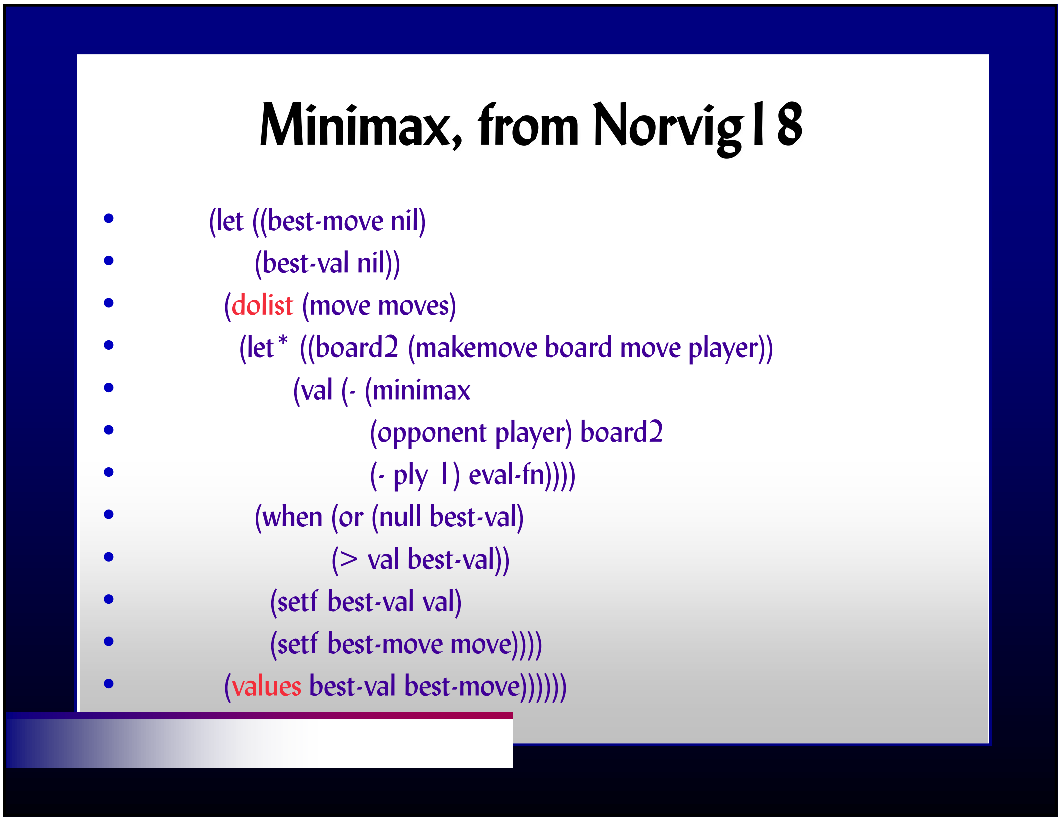
WIKIPEDIAS PSEUDOCODE

```
•function minimax(node, depth, maximizingPlayer) is
•   if depth = 0 or node is a terminal node then
•       return the heuristic value of node
•   if maximizingPlayer then
•       value := -8
•       for each child of node do
•           value := max(value, minimax(child, depth - 1, FALSE))
•       return value
•   else (* minimizing player *)
•       value := +8
•       for each child of node do
•           value := min(value, minimax(child, depth - 1, TRUE))
•       return value
•(* Initial call *)
•minimax(origin, depth, TRUE)
```



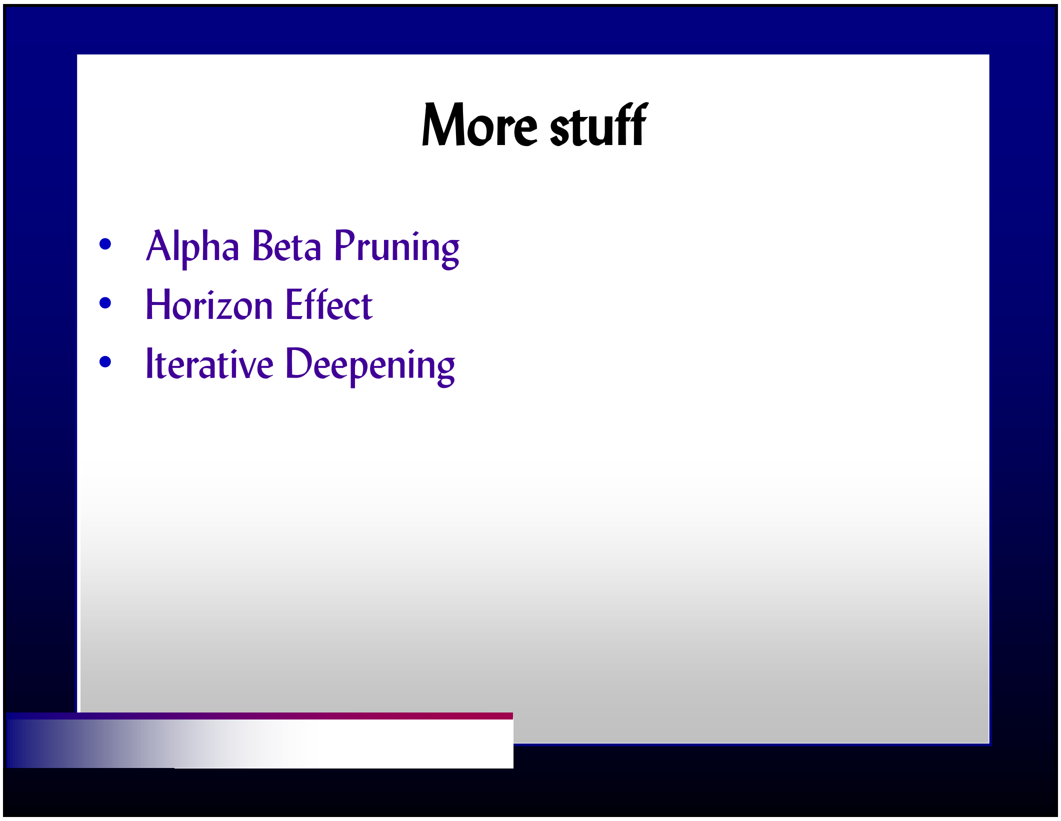
Minimax, from Norvig

```
•(defun minimax (player board ply eval-fn)
•  "Find the best move, for PLAYER, according to EVAL-FN,
•  searching PLY levels deep and backing up values."
•  (if (= ply 0)
•      (funcall eval-fn player board)
•      (let ((moves (legal-moves player board)))
•        (if (null moves)
•            (if (legal-moves (opponent player) board)
•                (- (minimax (opponent player) board
•                            (- ply 1) eval-fn))
•                (final-value player board))
•    ...
```



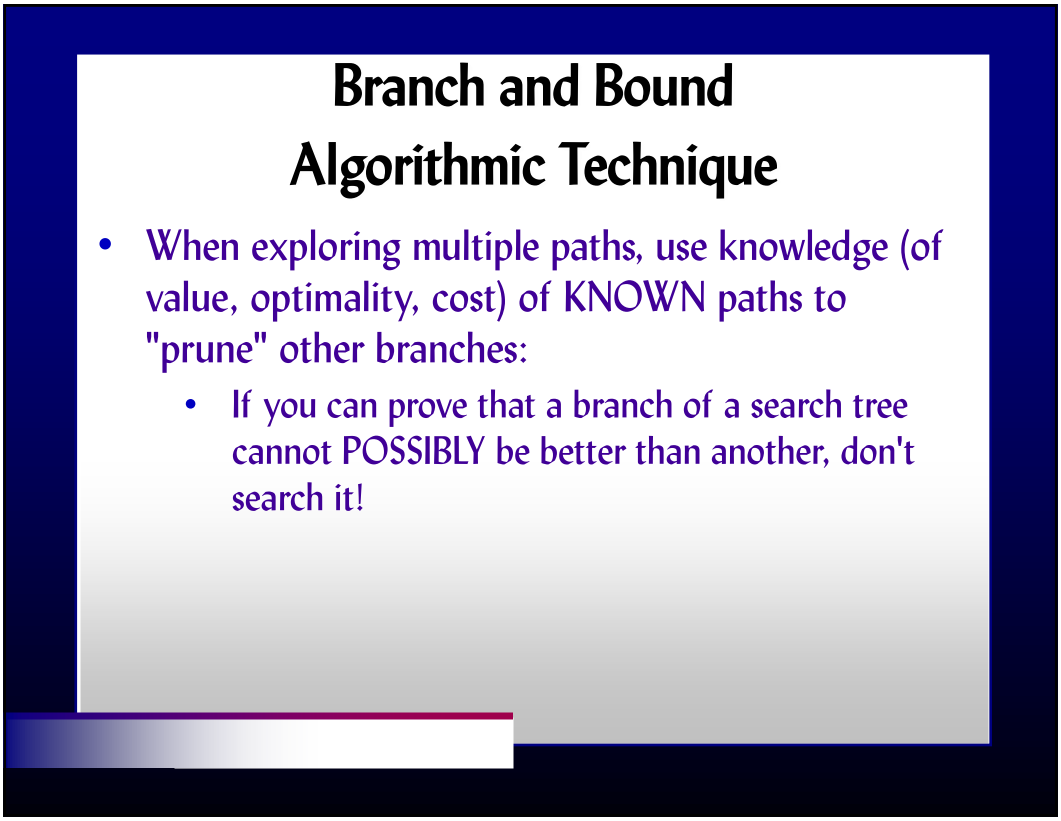
Minimax, from Norvig18

```
(let ((best-move nil)
      (best-val nil))
  (dolist (move moves)
    (let* ((board2 (makemove board move player))
           (val (- (minimax
                     (opponent player) board2
                     (- ply 1) eval-fn))))
      (when (or (null best-val)
                (> val best-val))
        (setf best-val val)
        (setf best-move move))))
  (values best-val best-move)))))
```



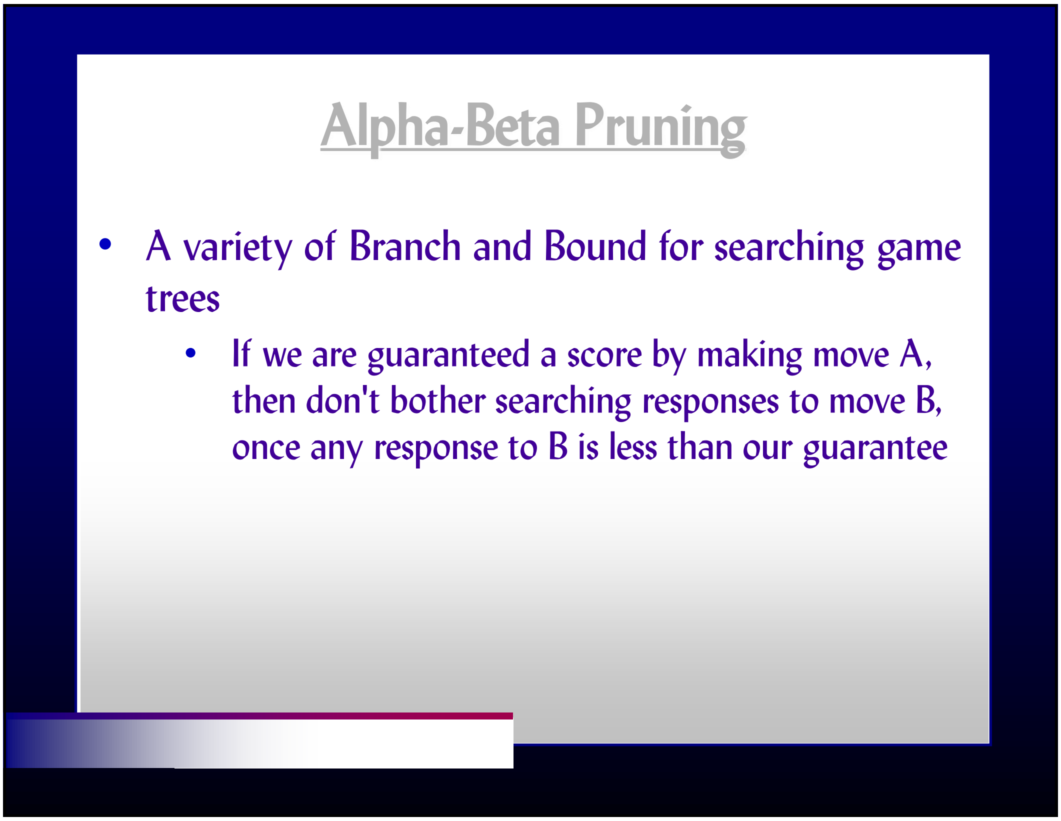
More stuff

- Alpha Beta Pruning
- Horizon Effect
- Iterative Deepening



Branch and Bound Algorithmic Technique

- When exploring multiple paths, use knowledge (of value, optimality, cost) of KNOWN paths to "prune" other branches:
  - If you can prove that a branch of a search tree cannot POSSIBLY be better than another, don't search it!

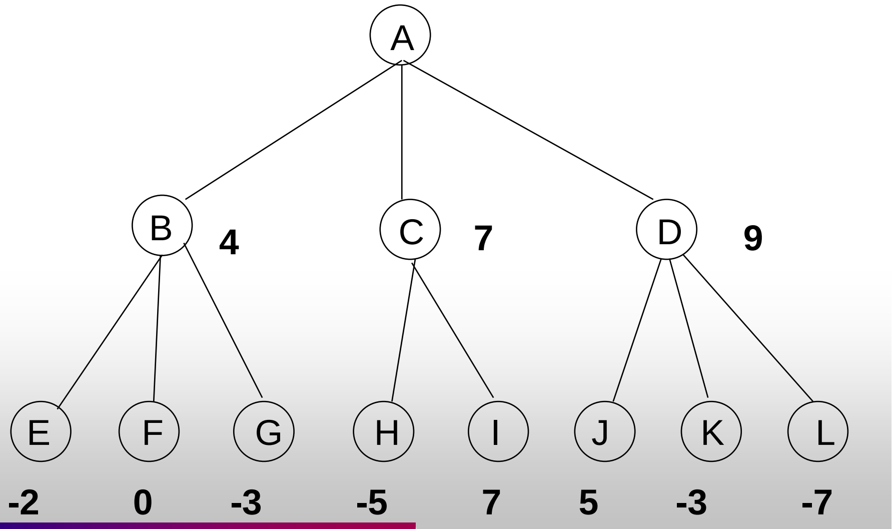


Alpha-Beta Pruning

- A variety of Branch and Bound for searching game trees
  - If we are guaranteed a score by making move A, then don't bother searching responses to move B, once any response to B is less than our guarantee

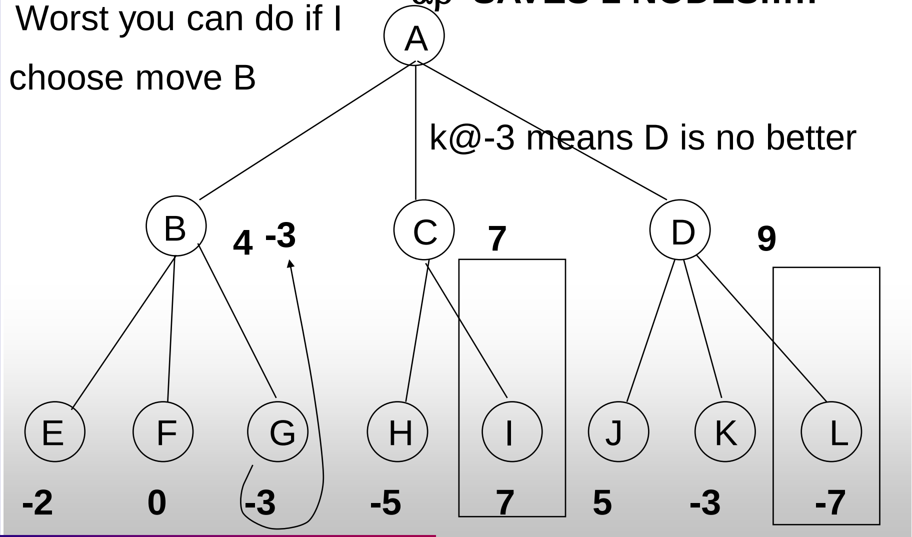
What are Alpha and Beta

- α ALPHA - Greatest Lower Bound on my score
  - (worst you can do to me)
- β BETA - Least Upper Bound on your score
  - (Best I can do against you)
    - These are used recursively in a flip-flop fashion



α= -3
Worst you can do if I choose move B

αβ SAVES 2 NODES!!!!!

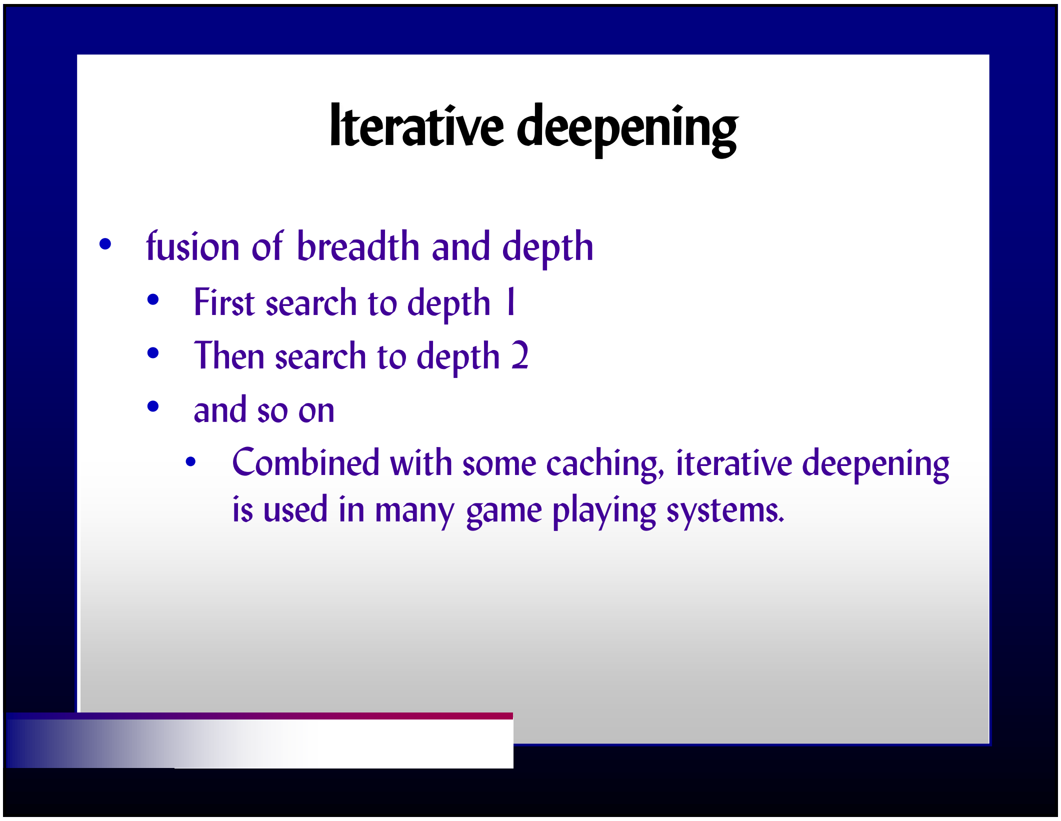


k@-3 means D is no better

H@-5 means C is worse than B

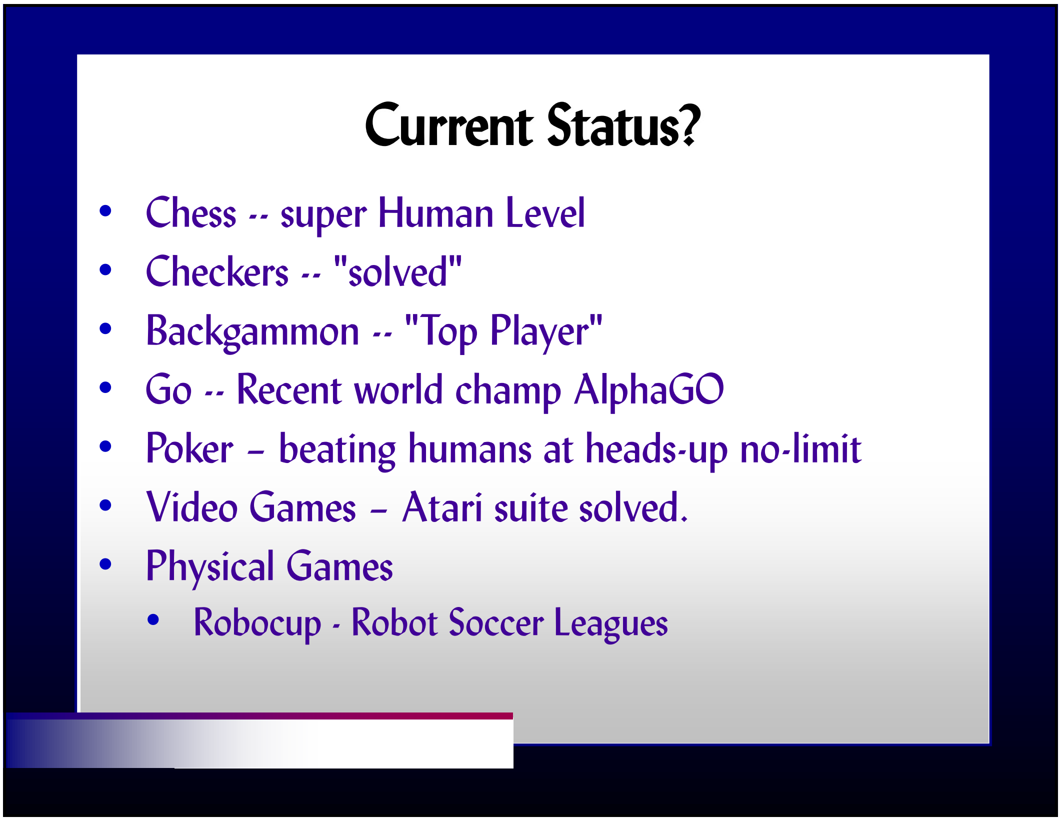
Gaming Complications

- Instability of Scoring Heuristic
  - In games with value exchange, the heuristics are very bumpy
  - *Make smoothing assumptions*
  - *search for "quiesence"*
- The Horizon Effect
  - No matter how deep you search, there may be a loss just beyond the horizon.
  - *Use secondary searching on a few final candidates*



Iterative deepening

- fusion of breadth and depth
  - First search to depth 1
  - Then search to depth 2
  - and so on
    - Combined with some caching, iterative deepening is used in many game playing systems.



Current Status?

- Chess -- super Human Level
- Checkers -- "solved"
- Backgammon -- "Top Player"
- Go -- Recent world champ AlphaGO
- Poker – beating humans at heads-up no-limit
- Video Games – Atari suite solved.
- Physical Games
  - Robocup - Robot Soccer Leagues