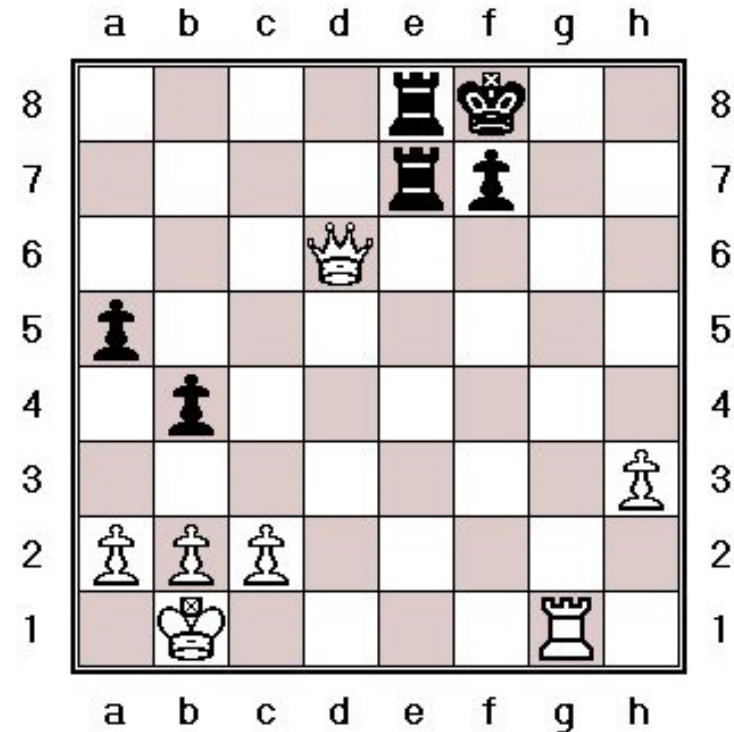


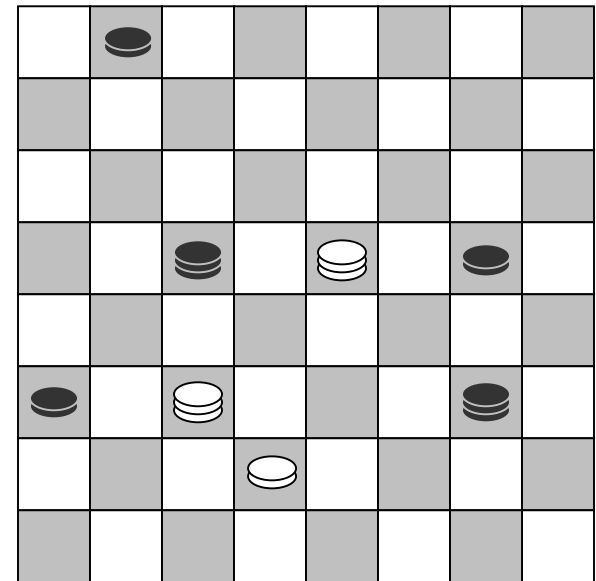
# Game Playing

- Game-playing is a form of searching that takes place in an adversarial domain
- Not concerned with solitaire-like games that do not involve an opponent
- Main question of game theory: What is the best move do make in the current situation?



# Assumptions

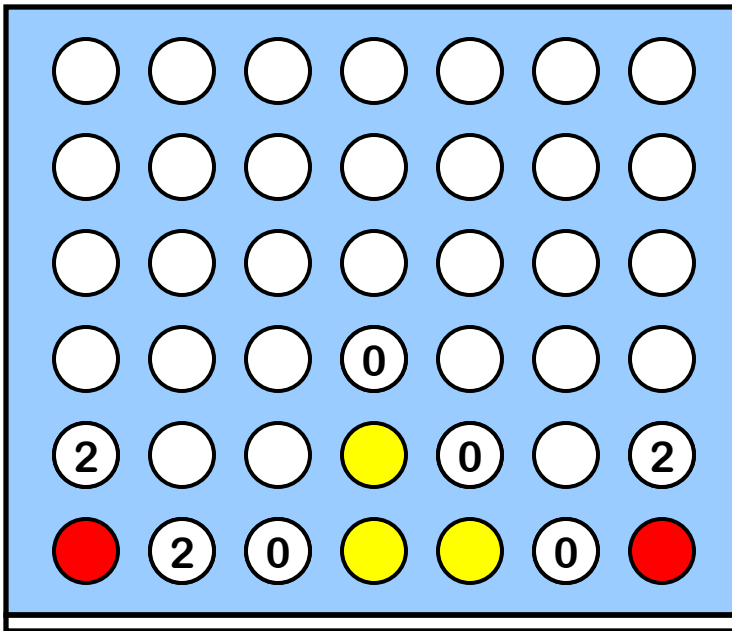
1. Domain can't be exhaustively explored
2. No **strictly dominating strategies**
3. Both the player's and opponent's strategies and positions are knowable
4. Both the player and the opponent are **rational**
5. Strategies are comprised of heuristics that can measure the "goodness" of any position with a numerical result
6. Game searches are pursuing a single goal
7. **Zero-sum** games being examined



↑ What move by white guarantees victory?

# Example: Connect Four

- Given the following game layout, what would the best move be for the red player?



## Game Rules:

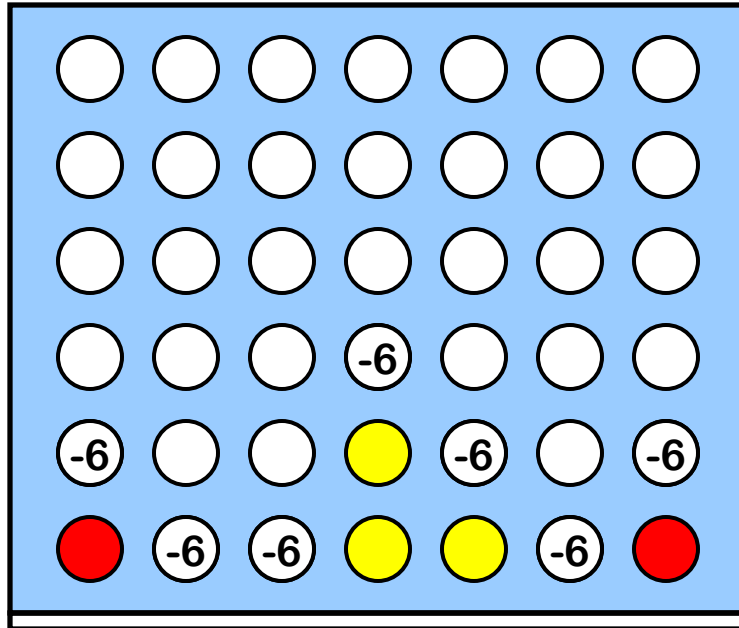
- Pieces alternate inserting pieces into a column, where it falls to the bottom-most unoccupied position
- First player with four tiles in a row in any direction wins

## Heuristic $h_1(s)$ :

Add 2 for every unblocked row of 2, 3 for every unblocked row of 3.

- A greedy best-first heuristic that ignores the opponent would choose columns 1, 2 or 7 to get closer to the goal

# Game Example (cont'd)



Heuristic  $h_2(s)$ :

- Add 2 for every unblocked row of 2; 3 for every unblocked row of 3.
- Subtract 2 for every row of 2 for the opponent, subtract 3 for every unblocked row of 3 for the opponent

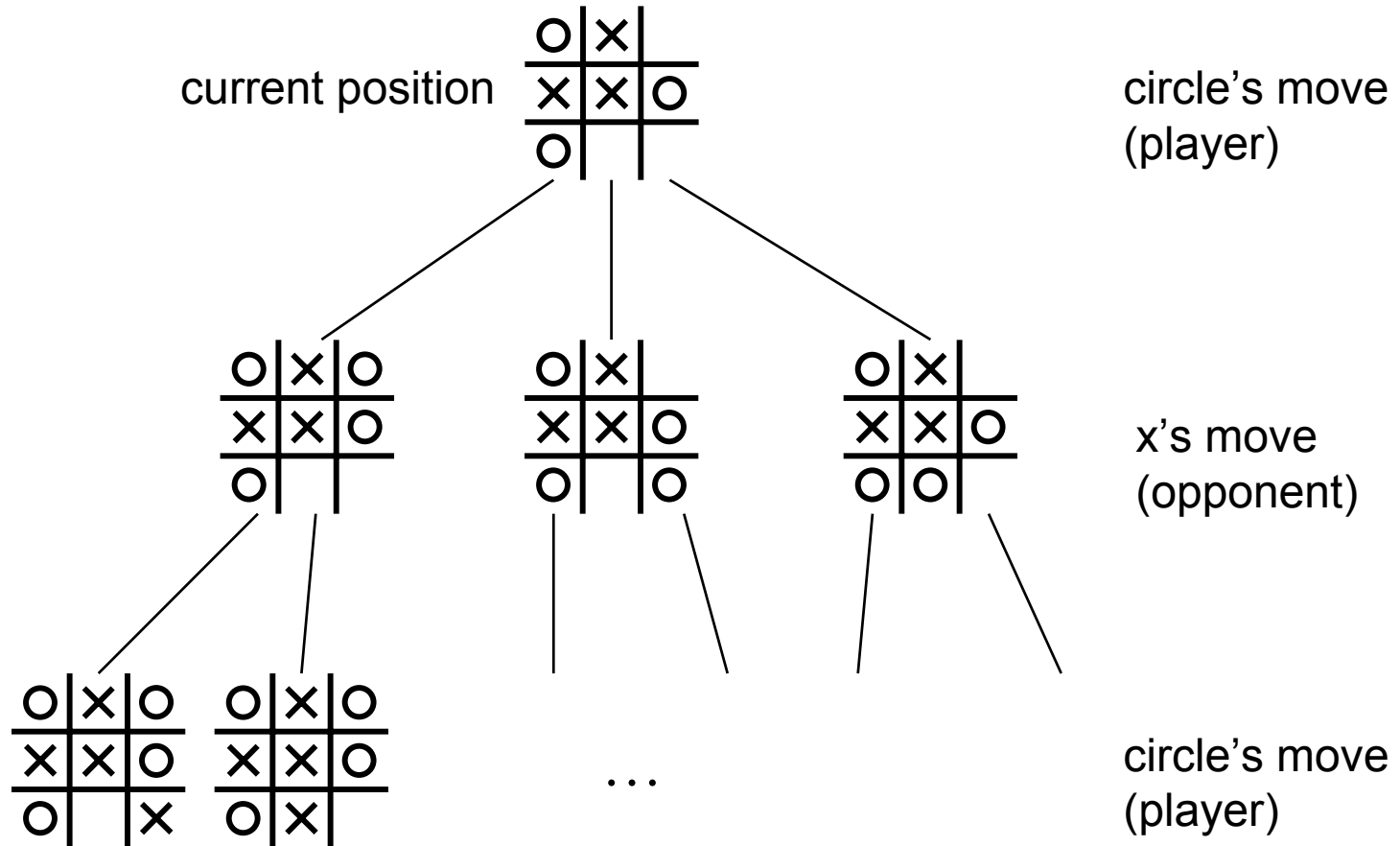
- Opponent's adversarial stance changes the interpretation of the problem. Players must account for the opponent trying to reduce the player's heuristic value in certain situations.
- The example heuristic implies that all positions are equal if one considers the heuristics for both players at the same time
  - Why isn't this an accurate assessment?

# Game Trees

- **Game trees** are used to reflect all two-player game scenarios
  - multi-player scenarios are possibly as well, and are an extension of two-player games
- Same as search trees, but take the opponent's goals into account as well
- Game trees form the foundation for all intelligent game-playing algorithms:
  - **Checkers**: AI can beat world champion
  - **Othello/Reversi**: Basic computer can beat best human player
  - **Go**: Amateur human can beat best computer player
  - **Chess**: Humans and computers still battling for top spot

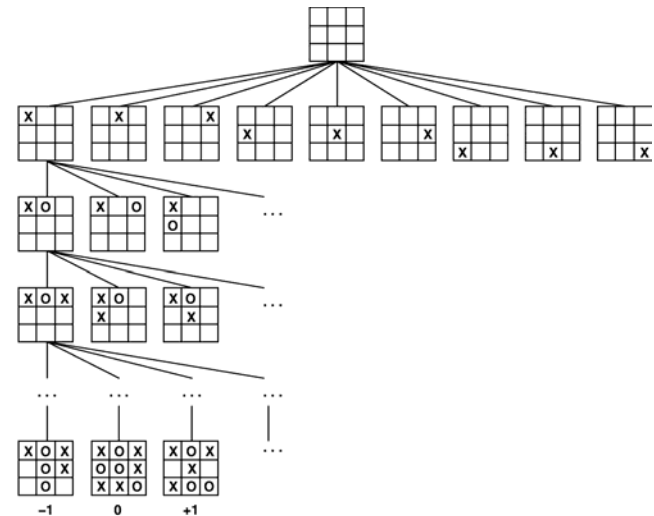
# Game Tree Example

- Tic-Tac-Toe:



# Branching in Game Trees

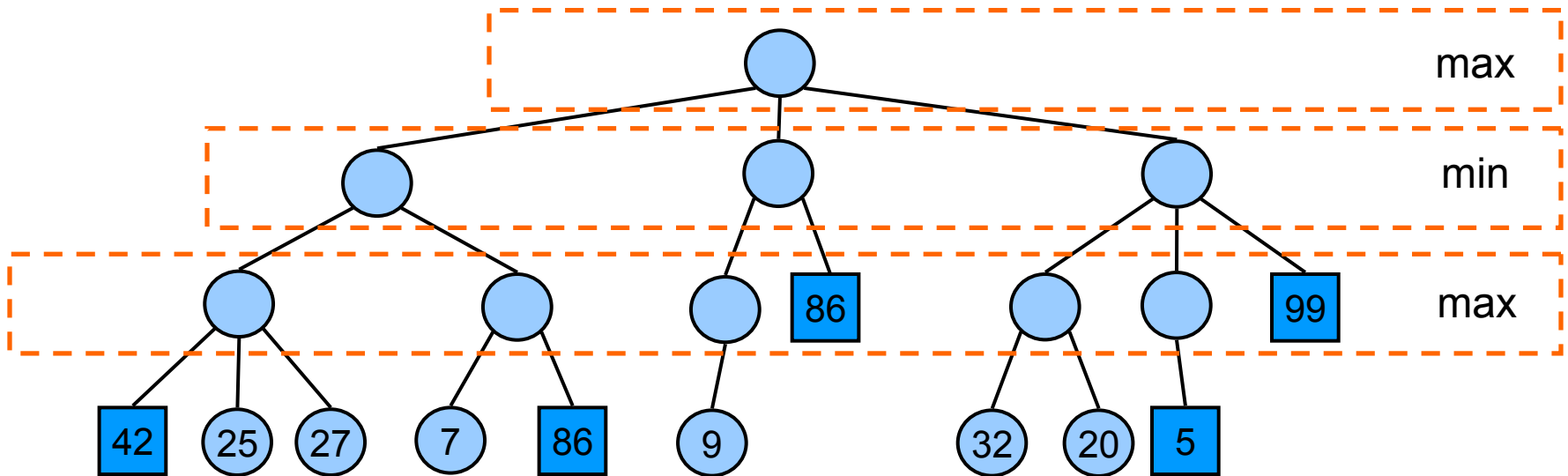
- Like any search tree, game tree branching can get out-of-hand very quickly
  - tic-tac-toe has 97,162 potential game positions to consider
  - chess has an average branching factor  $b = 42$ 
    - two moves on each side produces over 3 million possible positions!
- Solution:
  - look limited moves ahead
  - use good heuristic function
    - keep extensive databases
  - use **minimax** principle
  - prune search tree



# Minimax Principle

- Invented by von Neumann and Morgenstern in 1944 as part of game theory.
- Involves growing a game tree to the **search horizon**.
  - The **search horizon** is defined by the number of moves the computer looks ahead. If the computer look  $n$  moves ahead for itself and  $n - 1$  moves for the opponent, we say the computer is playing  $2n - 1$  **ply**.
- Within the tree bounded by the search horizon, apply the heuristic function to all leaves to calculate the utility of the position at each leaf.
- Idea behind **minimax** is that the AI player tries to maximize the utility while the opponent tries to minimize it.
- Note:
  - Leaf states are not necessarily end states
  - End states do not necessarily satisfy the goal conditions
  - Heuristic values at end states can vary

# Minimax Example

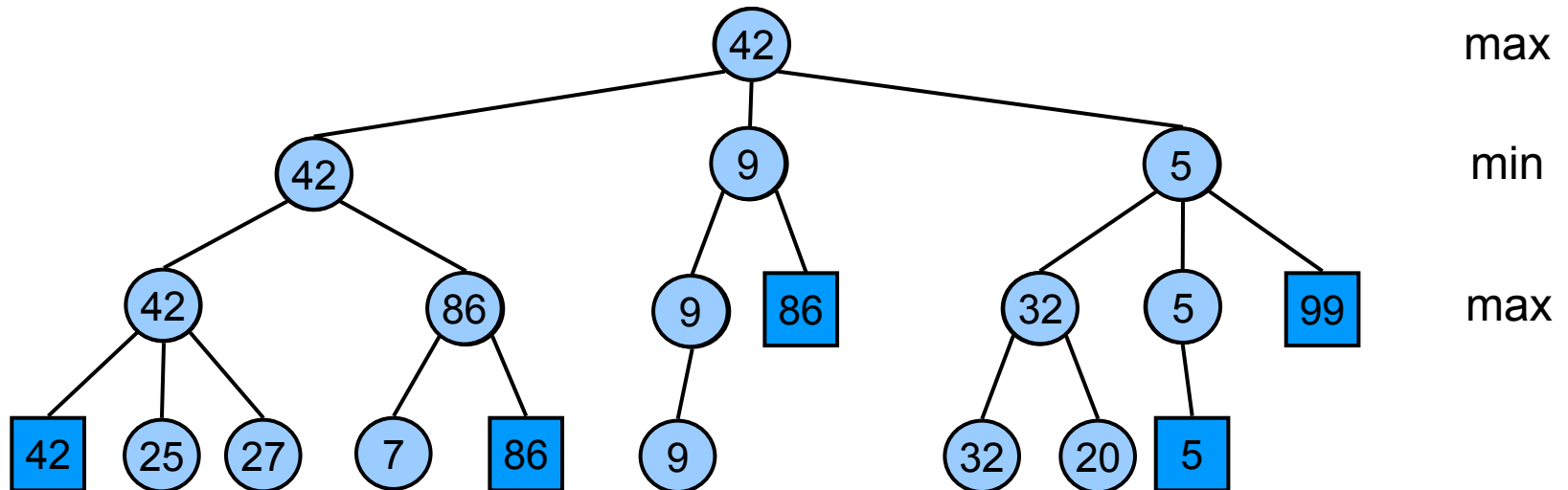


search horizon

- From root position, what branch should the AI player pursue to achieve the highest eventual utility?
  - Note: heuristic values are from the player's perspective, not the opponent's. It is assumed that the opponent wishes to minimize the player's utility at each min stage.

# Minimax Example (cont'd)

- Minimax algorithm:
  - from bottom level to top, fill in node with either the maximum of its children (on max levels) or the minimum of its children (on min levels)



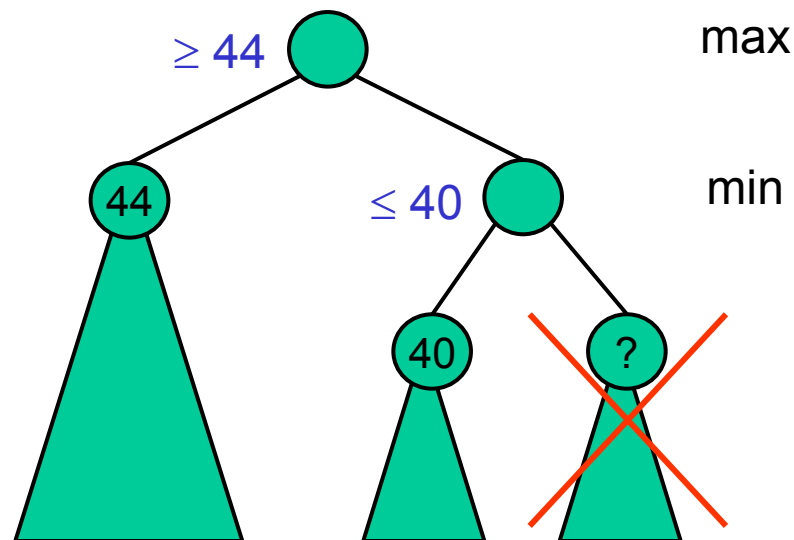
→ Best path is left-most branch

# Minimax & Pruning

- Assumptions:
  - Heuristic is known by both player and opponent
  - both will make the most sensible move, given their goals
- Minimax alone will not solve the search problem
  - need reduction in search space in order to increase chances of arriving at the goal
- **Pruning** is a major technique used to reduce the branch density and speed up the search
  - e.g. prior knowledge about certain positions can allow any further positions to be discounted
- **$\alpha\beta$ -pruning** can cut down the number of nodes searched *without losing information*

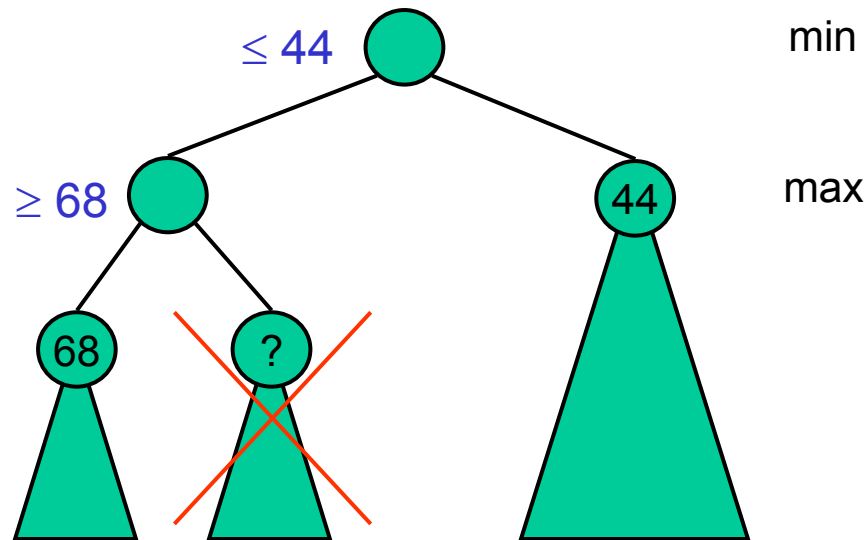
# $\alpha$ -Pruning

- When exploring nodes in game tree, assumption is a limited depth-first search, from left to right.
- $\alpha$ -pruning keys on the max level, and discards any branches that won't affect the max level value.
- Example:



# $\beta$ -Pruning

- $\beta$ -pruning is the same as  $\alpha$ -pruning, only from the perspective of the min levels



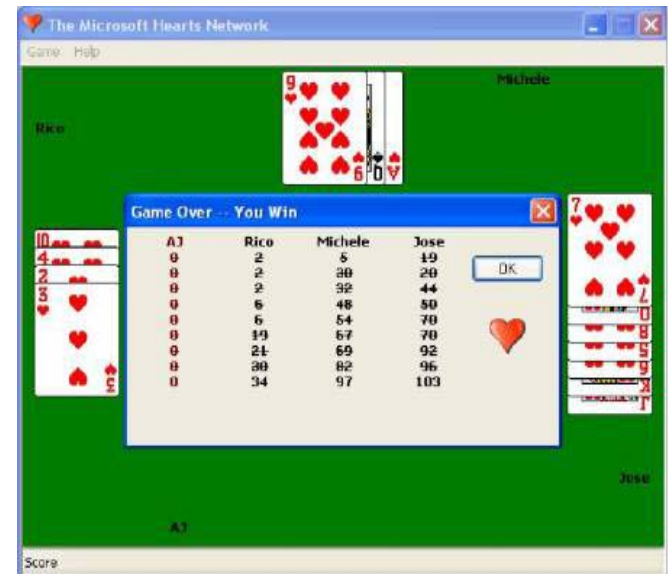
# $\alpha\beta$ -Pruning (cont'd)

- Before a node and its subtree can be discarded, the algorithm requires tentative values for parent and grandparent nodes in tree
- Complexity reduction:
  - $\alpha\beta$ -pruning reduces searching to  $O(n^{1/2})$
  - search is able to explore twice the depth of a regular search
    - very helpful with real-time game applications



# Strategies

- Game tree examples so far have assumed that both players are operating with the same **strategy** when making decisions
- Strategies are collections of heuristics that unify to promote a common goal
  - e.g. more conservative, more aggressive, taking advantage of certain sections of the board/field, etc.
  - strategies can sometimes work in tandem, but often have mutually exclusive goals
- **Mixed strategies** produce heuristics that are a weighted combination of two or more strategies



# Strategies (cont'd)

- As the results of a mixed strategy are observed during the progress of the game, the player may adjust the weights both during and between games
- Since an opponent's heuristics cannot be accurately predicted given mixed strategies, the best approach for making minimax calculations is to:
  1. assume a mixed strategy identical to one's own
  2. adjust the believed weights of the opponent's strategies according to the strategies that correlate with the opponent's past moves (**belief probability**)
- Strategy adjustment will eventually reach equilibrium
  - **Nash equilibrium**: a strategy state where no player can stand to benefit from adjusting their strategy while other players' strategies remain unchanged

# Multi-Player Games

- To extend the game tree idea to multiple players, a few basic adjustments are necessary:
  - the collective moves of adversarial players are calculated and treated as a single min level in the game tree
  - co-operative players' moves are incorporated as a change in the state of the game, without incorporating their effect into either the min or the max levels
  - if two or more players coordinate:
    - their moves have to be calculated together, if they are opponents
    - their moves have to be incorporated into the AI player's max level, if the coordination is with the player (collective goals)

# Search Enhancements

- Other techniques exist for speeding up search through game space
- **Transposition tables**
  - tables that record past positions, to avoid searching previously-explored subtrees
  - also used to eliminate subtrees that are permutations of other positions
    - e.g. Tic-tac-toe: initial branching reduces from  $b=9$  to  $b=3$
  - Disadvantage: most effective with iterative deepening search, which undermines use of  $\alpha\beta$ -pruning
  - must be careful not to equate two states whose positions are similar but situations are different
    - e.g. castling in chess, inventory in first-person shooters

# Search Enhancements (cont'd)

- **Opening book**
  - games with set initial positions tend to have predetermined patterns (i.e. chess)
  - opening books reduce the search space in initial situations by performing one of a set of opening actions
- **Endgames / Killer moves**
  - certain situations that match recognized patterns can trigger a series of actions that guarantee an increase in the utility of the game state
  - endgames in particular lead to outcomes that guarantee a win
- **Variable depth**
  - the search horizon can be adjusted if a particular path requires more exploration (i.e. to realize a sub-goal)

# Computer Chess Ratings Throughout History

