CS 1571 Introduction to AI
Lecture 11a

Adversarial search

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square

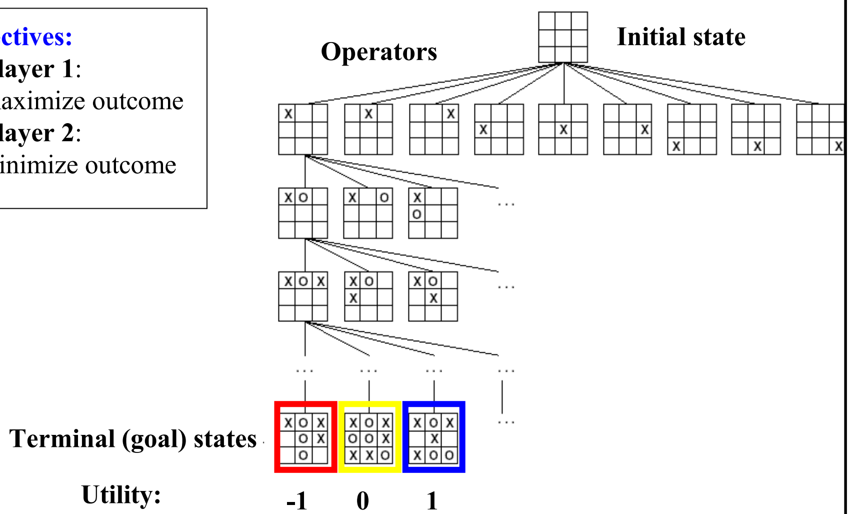
Game search problem

- **Game problem formulation:**
  - **Initial state:** initial board position + info whose move it is
  - **Operators:** legal moves a player can make
  - **Goal (terminal test):** determines when the game is over
  - **Utility (payoff) function**: measures the outcome of the game and its desirability

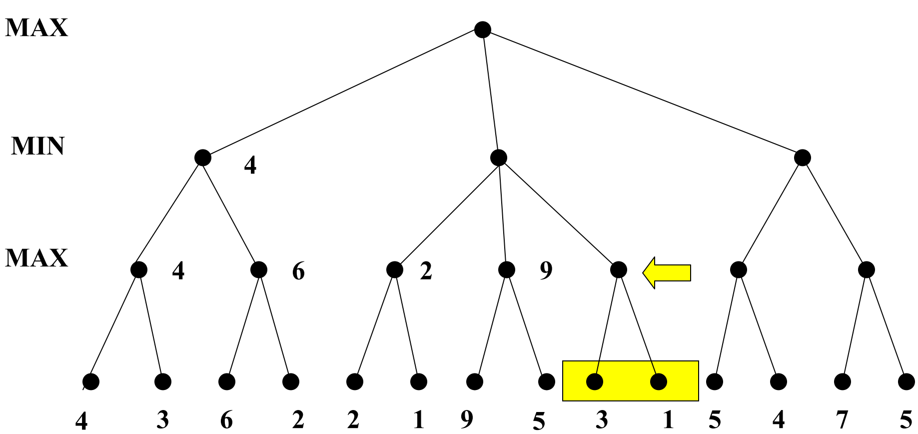
- **Search objective:**
  - find the sequence of player's decisions (moves) maximizing its utility (payoff)
  - Consider the opponent's moves and their utility

Game problem formulation (Tic-tac-toe)

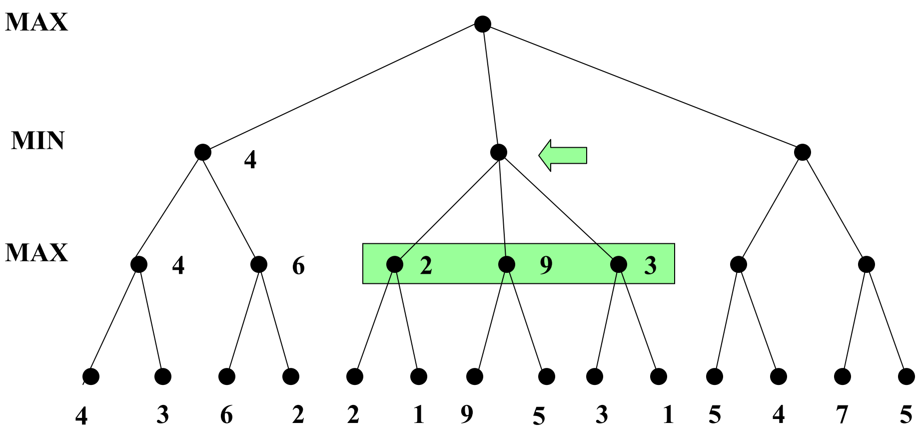
Objectives:
- **Player 1**:
  maximize outcome
- **Player 2**:
  minimize outcome



Minimax algorithm. Example

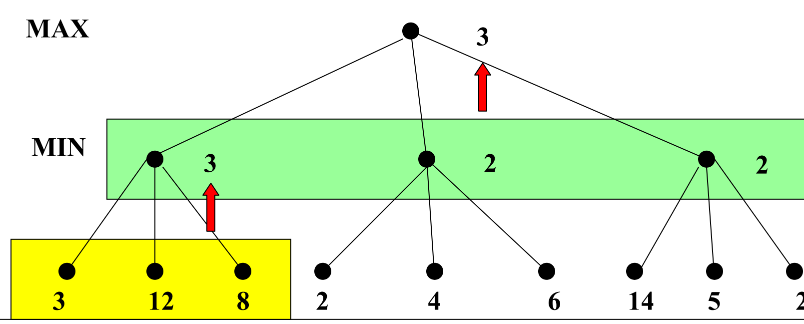


Minimax algorithm. Example



Minimax algorithm

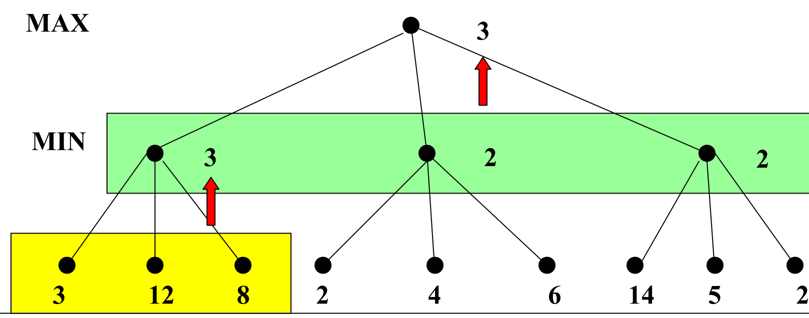
What assumption does the minimax algorithm make about the opponent?



Minimax algorithm

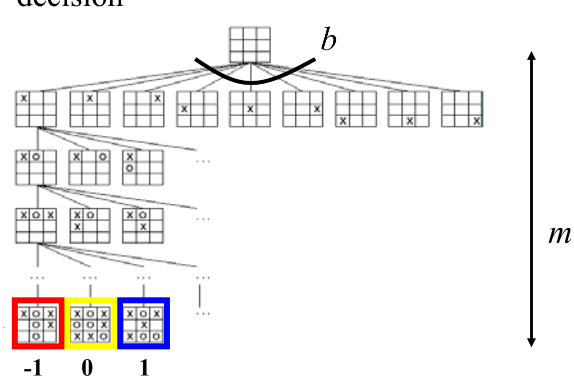
What assumption does the minimax algorithm make about the opponent?

- the opponent is rational; we assume **the best opponent's response**



Complexity of the minimax algorithm

- We need to explore the complete game tree before making the decision



Complexity:

$O(b^m)$

- Impossible for large games
  - Chess: 35 operators, game can have 50 or more moves

Solution to the complexity problem

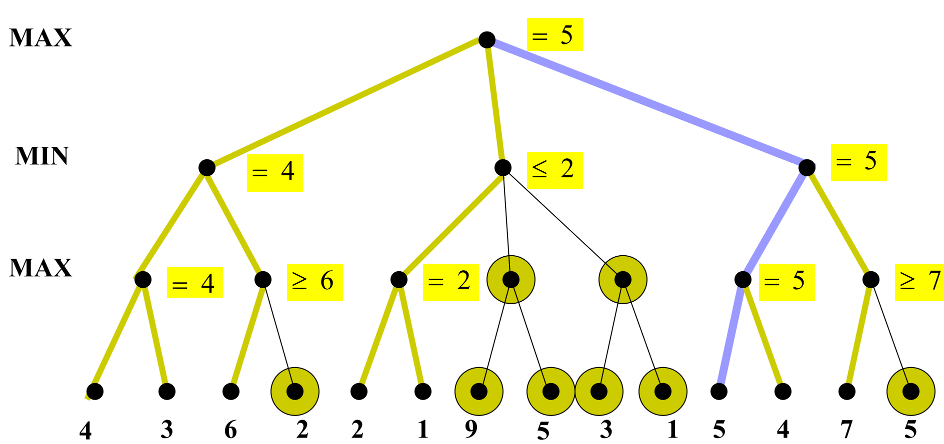
Two solutions:

1. **Dynamic pruning of redundant branches** of the search tree
   - identify a provably suboptimal branch of the search tree before it is fully explored
   - Eliminate the suboptimal branch

   Procedure: Alpha-Beta pruning

2. **Early cutoff of the search tree**
   - uses imperfect minimax value estimate of non-terminal states (positions)

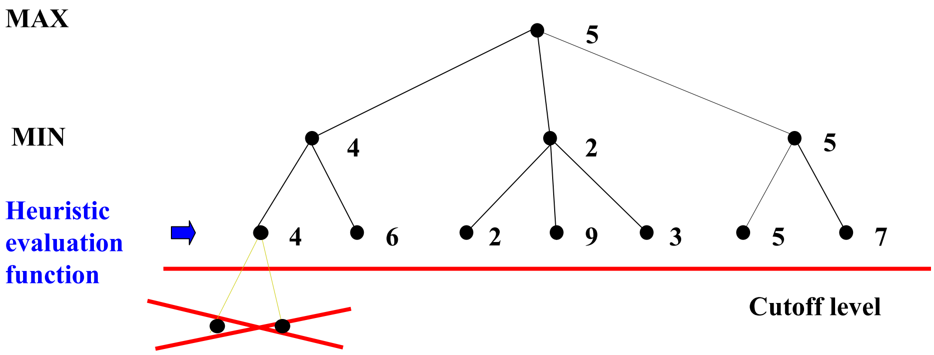
Alpha beta pruning. Example



nodes that were never explored !!!

Using minimax value estimates

- **Idea:**
  - Cutoff the search tree before the terminal state is reached
  - Use imperfect estimate of the minimax value at the leaves
    - (Heuristic) evaluation function



Design of evaluation functions

- **Heuristic estimate** of the value for a sub-tree
- **Examples of a heuristic functions**:
  - **Material advantage in chess, checkers**
    - Gives a value to every piece on the board, its position and combines them
  - More general **feature-based evaluation function**
    - Typically a linear evaluation function:

$$f(s) = f_1(s)w_1 + f_2(s)w_2 + \dots f_k(s)w_k$$

$f_i(s)$ - a feature of a state s

w_i - feature weight

Further extensions to real games

- Play a restricted game: a restricted set of moves is considered under **the cutoff level** to reduce branching and improve the evaluation function
  - E.g., consider only the capture moves in chess