# Adversarial Search and Games

**Dr. Fayyaz ul Amir Afsar Minhas**
PIEAS Biomedical Informatics Research Lab
Department of Computer and Information Sciences
Pakistan Institute of Engineering & Applied Sciences
PO Nilore, Islamabad, Pakistan
http://faculty.pieas.edu.pk/fayyaz/

# Outline

- Introduction to Adversarial Search Problems
- Optimal Decision in Games
  - Minimax Algorithm
- Alpha Beta Pruning
- State of the Art Game Programs

# Adversarial Search Problems

- In presence of multiple agents the unpredictability of other agents can introduce many possible contingencies into the problem solving process
- Cooperation and Competition
- Game Theory: Deals with multiple agent environments as a game provided that the impact of each agent on the others is significant
- Competitive environments in which the agent’s goals are in conflict give rise to ASPs (games)

# Examples

- Games
  - Chess
  - Othello
  - Tic-Tac-Toe

# Prisoner’s Dilemma

- A typical example from Game Theory

| | Prisoner B Stays Silent | Prisoner B Betrays |
|---|---|---|
| **Prisoner A Stays Silent** | Each serves six months | Prisoner A serves ten years<br>Prisoner B goes free |
| **Prisoner A Betrays** | Prisoner A goes free<br>Prisoner B serves ten years | Each serves five years |

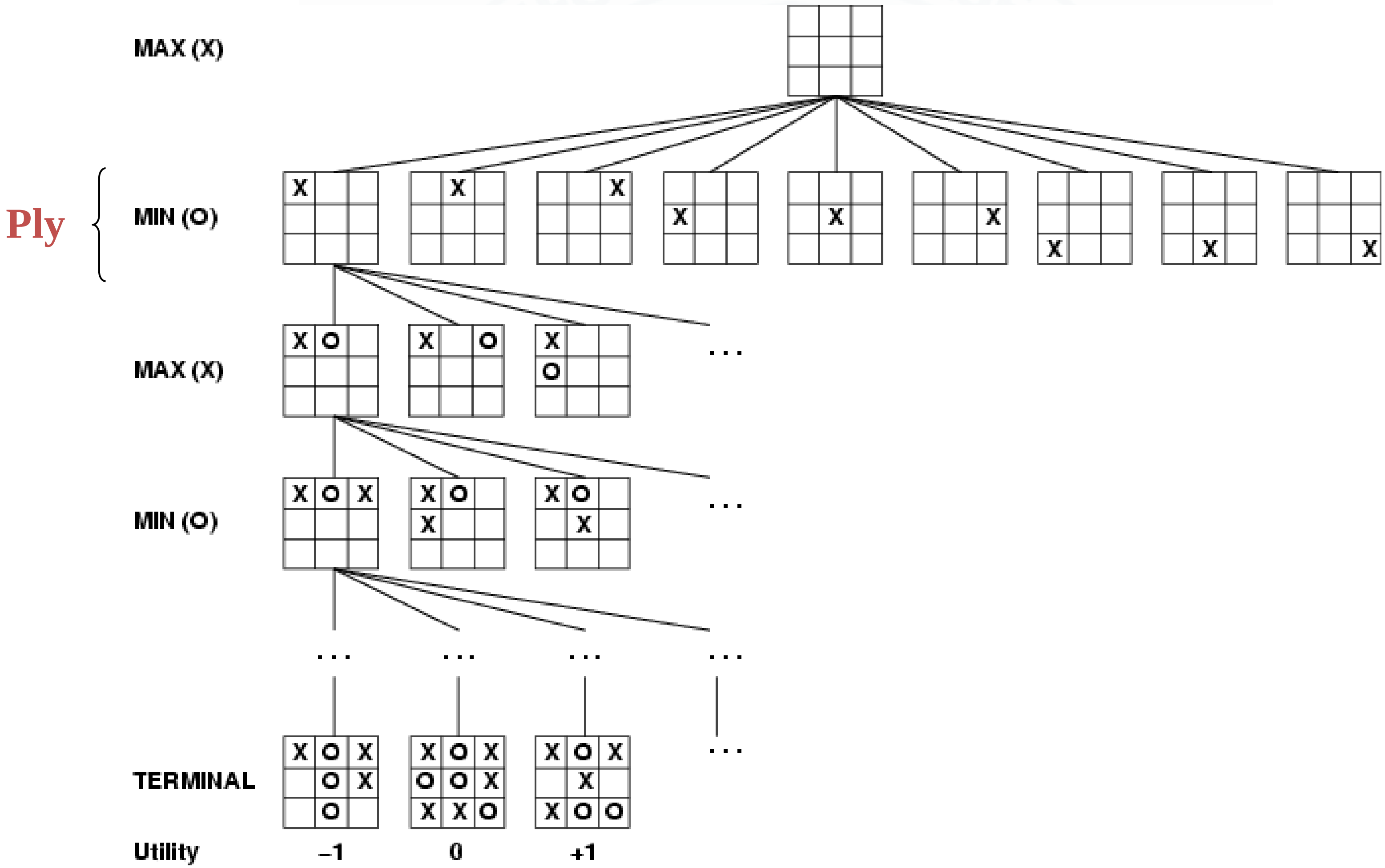
# NIM

- A number of tokens are placed on a table between the two opponents; at each move the player must divide a pile of tokens into two non-empty piles of different sizes.
  - For example, 6 tokens can be divided into piles of 5 & 1 or 4 & 2 but not 3 & 3.
- The first player who can no longer make a move loses the game.
- The utility function assigns a value of +1 when MAX is the winner and 0 otherwise.

# Optimal Decision in Games

- A game represented as an ASP has the following components
  - Initial State
    - Includes the board position and identifies the player
  - Successor Function
    - Which generates a list of (move, state) pairs each indicating a legal move and the resulting state
  - Terminal Test
    - Which determines when the game is over
    - States where the game has ended are called terminal states
  - Utility Function
    - Assigns a numeric value to the terminal state

# Game Tree

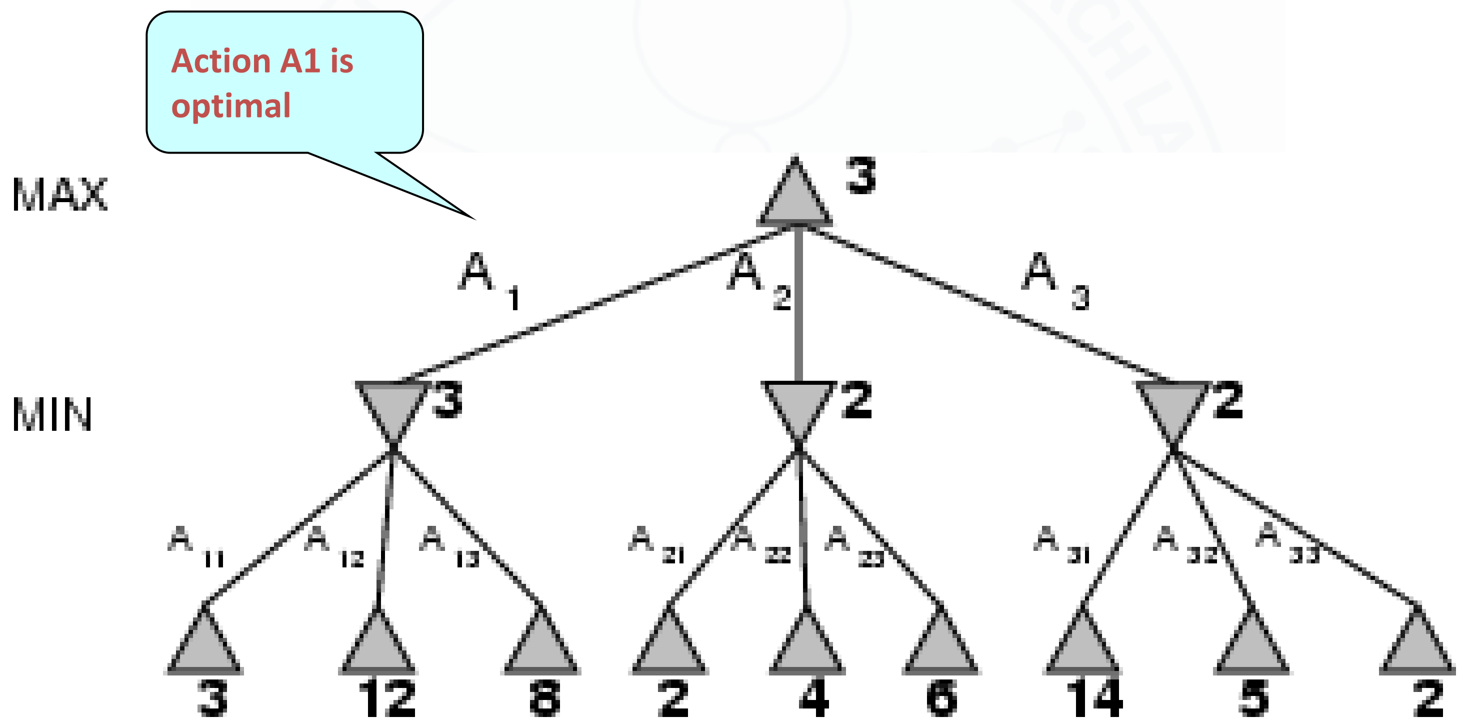


# Optimal Strategies

- MAX must find a contingent strategy in relation to MIN's actions
- The optimal strategy can be determined by examining the minimax value of each node

$$MINIMAX-VALUE(n)=\begin{cases} UTILITY(n) & \text{if n is a terminal node} \\ \max_{s\in Successors(n)} MINIMAX-VALUE(s) & \text{if n is a MAX node} \\ \min_{s\in Successors(n)} MINIMAX-VALUE(s) & \text{if n is a MIN node} \end{cases}$$

# Optimal Strategies…



# Optimal Strategy…

- Minimax Decision
  - Maximizes the worst case outcome for Max
  - What if MIN does not play optimally?
    - MAX will do even better

# MINIMAX Pseudocode

```
function MINIMAX-DECISION(state) returns an action
   return argmax_{a in ACTIONS(s)} MIN-VALUE(RESULT(state,a))
```

---

```
function MAX-VALUE(state) returns a utility value
   if TERMINAL-TEST(state) then return UTILITY(state)
   v ← −∞
   for each a in ACTIONS(state) do
       v ← MAX(v,MIN-VALUE(RESULT(s,a))
   return v
```

---

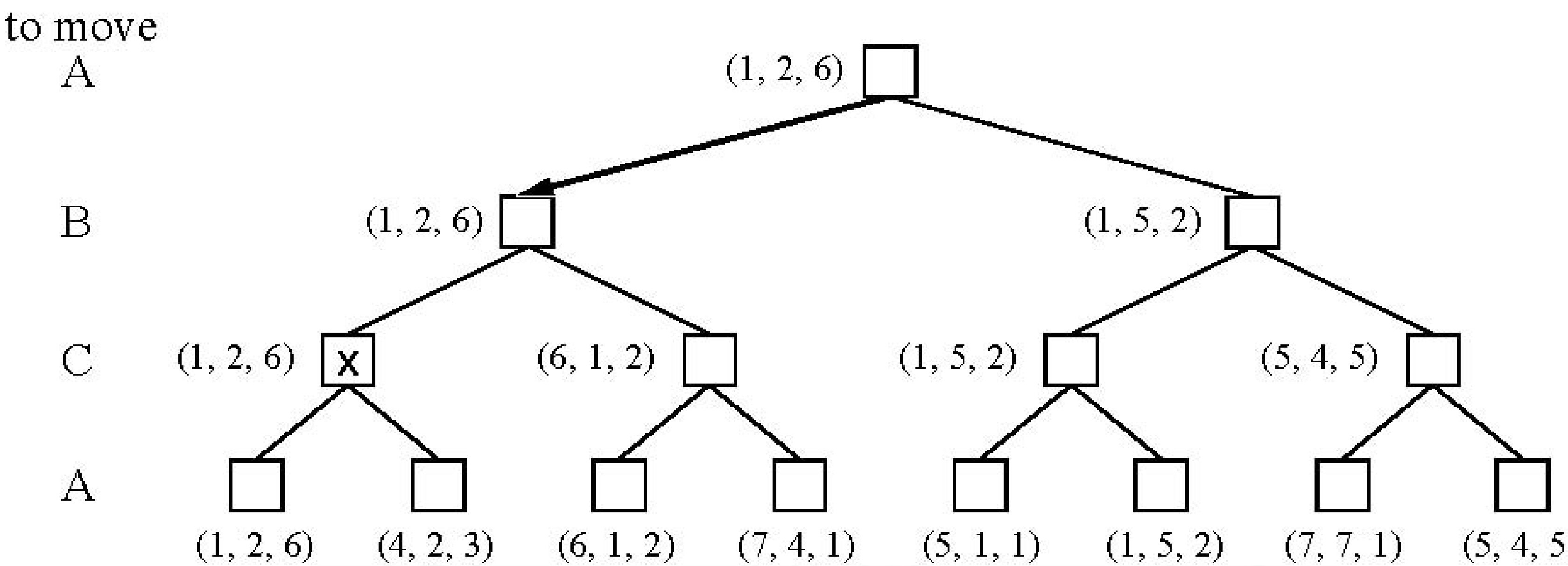
```
function MIN-VALUE(state) returns a utility value
   if TERMINAL-TEST(state) then return UTILITY(state)
   v ← ∞
   for each a in ACTIONS(state) do
       v ← MIN(v,MAX-VALUE(RESULT(s,a))
   return v
```

# Properties of minimax

- Complete? Yes (if tree is finite)
- Optimal? Yes (against an optimal opponent)
- Time complexity? $O(b^m)$
- Space complexity? $O(bm)$ (depth-first exploration)

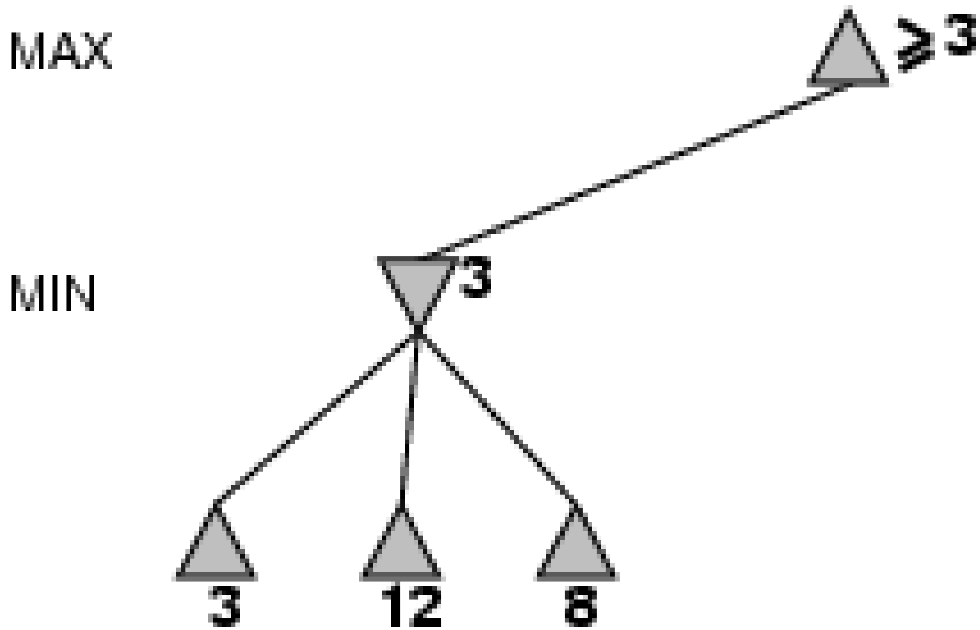
- For chess, $b \approx 35$, $m \approx 100$ for "reasonable" games
  → exact solution completely infeasible

# Optimal Decision in Multiplayer Games

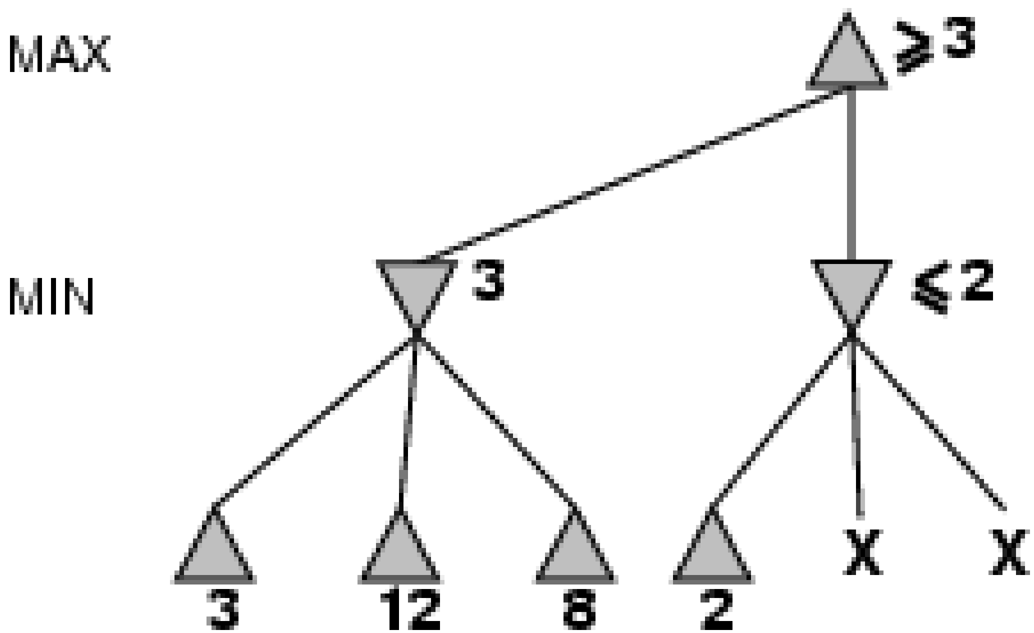



- Alliances can be a natural consequence of optimal strategies for each player in a multiplayer game
- If the game is not zero-sum, then collaboration can also occur with just two players

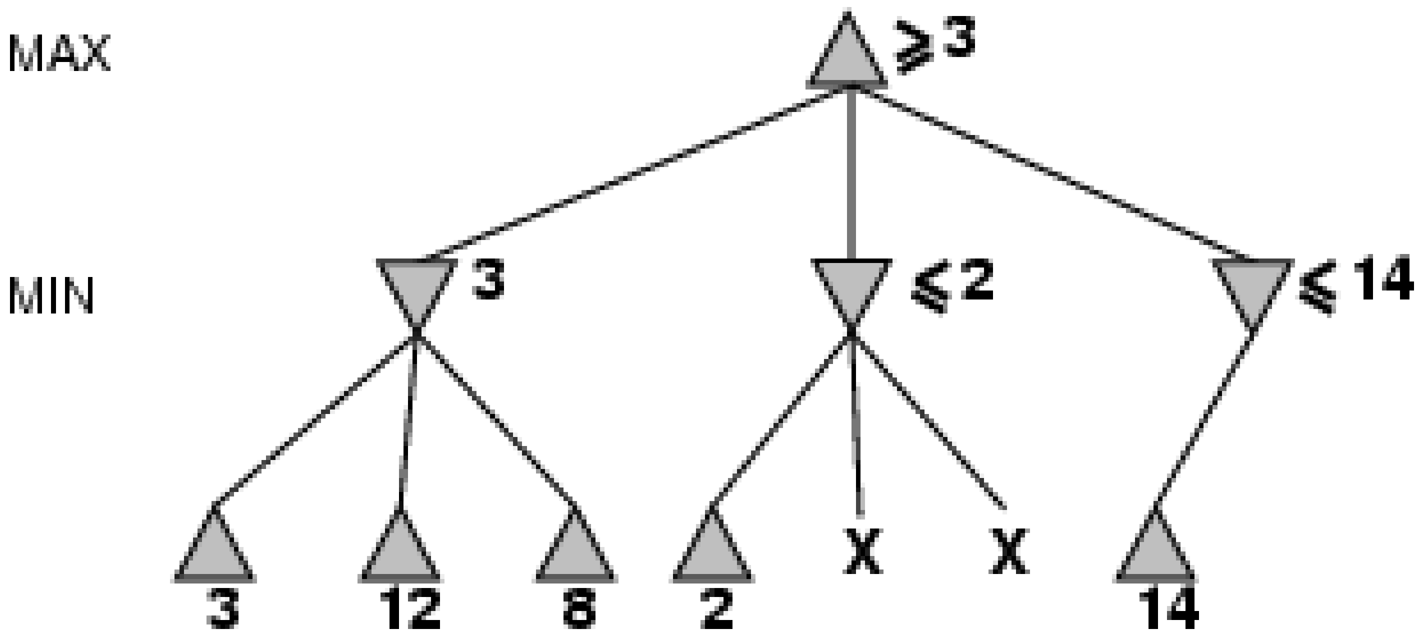
# α-β pruning example



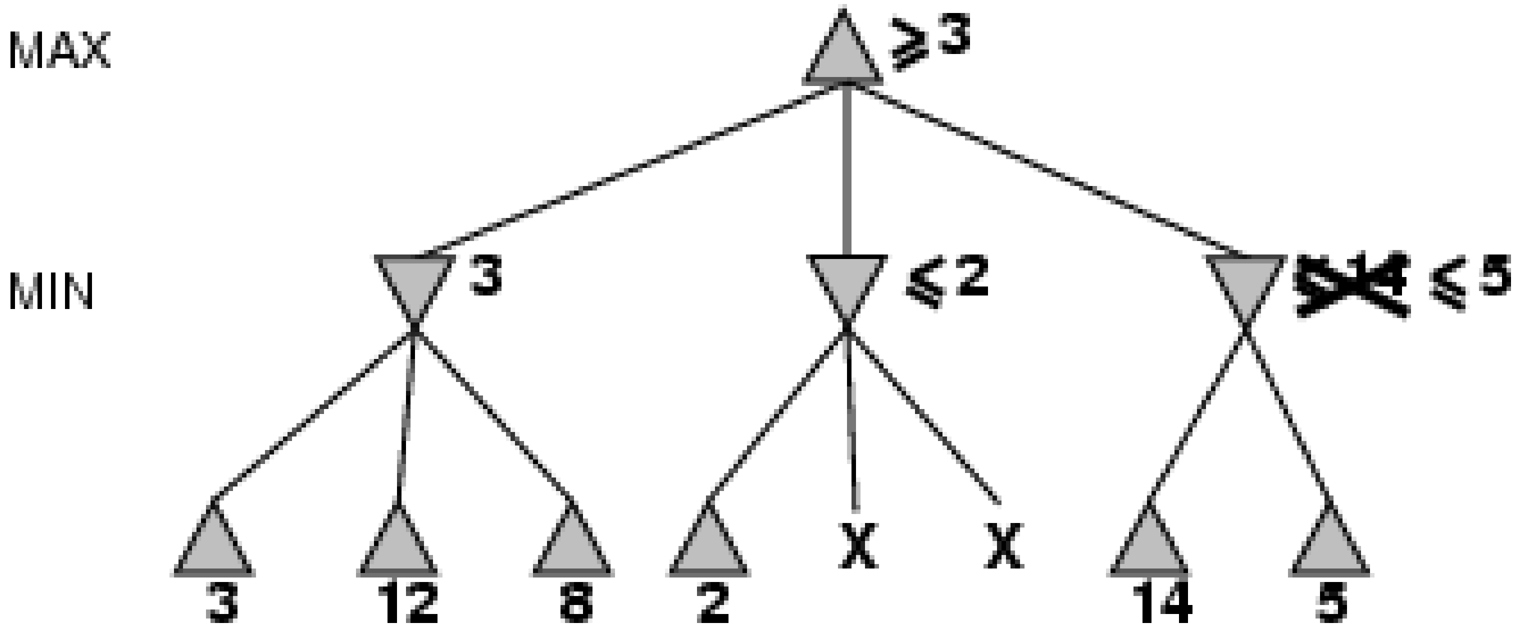
# α-β pruning example



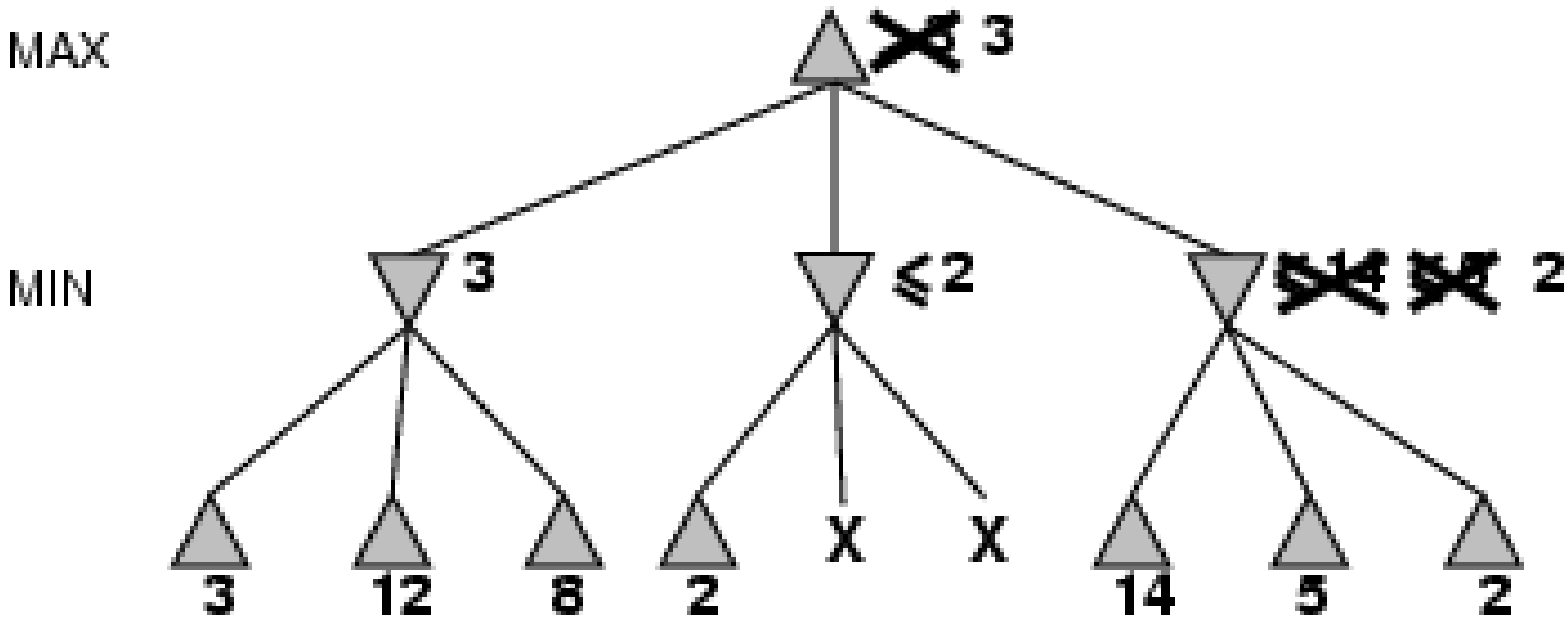
# α-β pruning example



# α-β pruning example



# α-β pruning example



# Properties of α-β

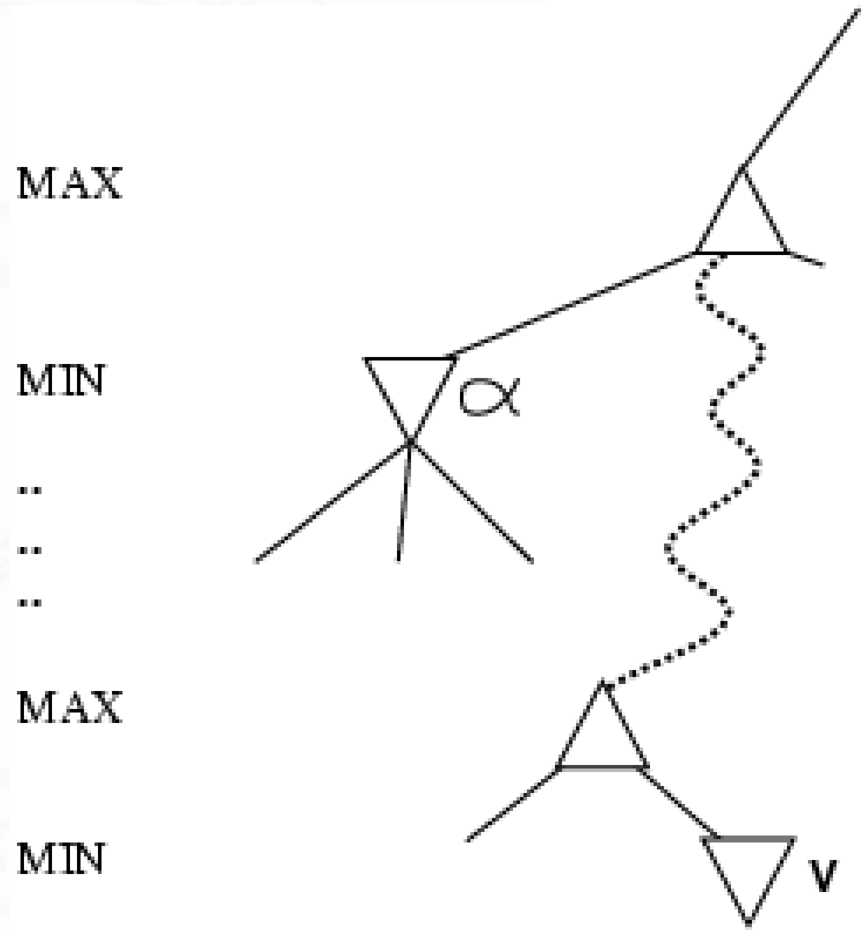
- Pruning does not affect final result
- Good move ordering improves effectiveness of pruning
- With perfect ordering, time complexity = $O(b^{m/2})$

  → doubles depth of search

# Why is it called α-β?

- α is the value of the best (i.e., highest-value) choice found so far at any choice point along the path for *max*
- If *v* is worse than α, *max* will avoid it

  → prune that branch
- Define β similarly for *min*



# The α-β algorithm

```
function ALPHA-BETA-SEARCH(state) returns an action
  inputs: state, current state in game

  v ← MAX-VALUE(state, −∞, +∞)
  return the action in SUCCESSORS(state) with value v
```

---

```
function MAX-VALUE(state, α, β) returns a utility value
  inputs: state, current state in game
          α, the value of the best alternative for  MAX along the path to state
          β, the value of the best alternative for  MIN along the path to state

  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← −∞
  for a, s in SUCCESSORS(state) do
    v ← MAX(v, MIN-VALUE(s, α, β))
    if v ≥ β then return v
    α ← MAX(α, v)
  return v
```

# The α-β algorithm

```
function MIN-VALUE(state, α, β) returns a utility value
   inputs: state, current state in game
           α, the value of the best alternative for  MAX along the path to state
           β, the value of the best alternative for  MIN along the path to state

   if TERMINAL-TEST(state) then return UTILITY(state)
   v ← +∞
   for a, s in SUCCESSORS(state) do
      v ← MIN(v, MAX-VALUE(s, α, β))
      if v ≤ α then return v
      β ← MIN(β, v)
   return v
```

# Resource limits

Suppose we have 100 secs, explore $10^4$ nodes/sec

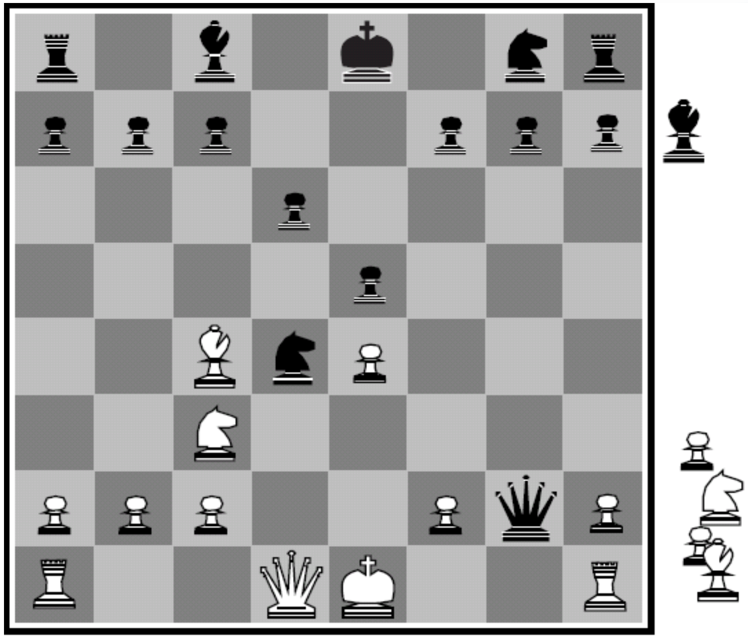
→ $10^6$ nodes per move

Standard approach:

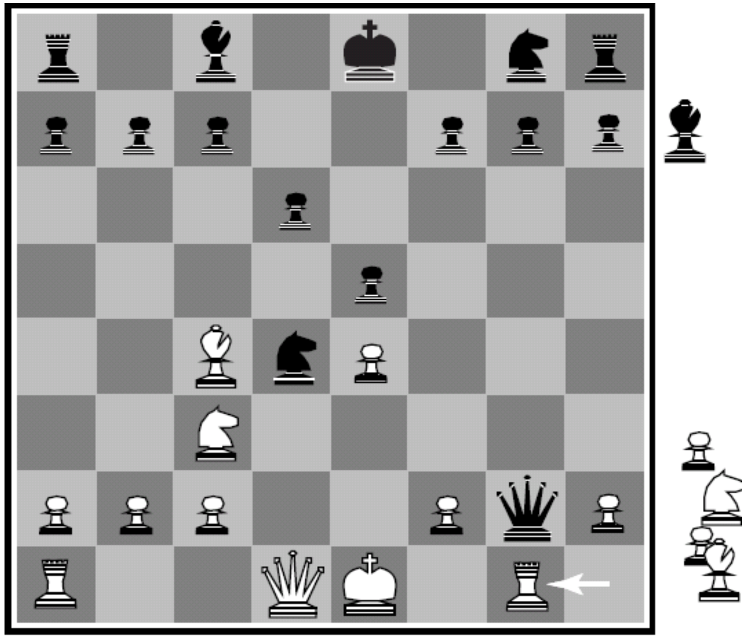
- cutoff test:
  - e.g., depth limit (perhaps add quiescence search)
  - Cutoff should only be applied to Quiescent (Steady) Positions that do not exhibit wild swings in value in the near future
- evaluation function

  = estimated desirability of position

# Quiescent Search



(a) White to move

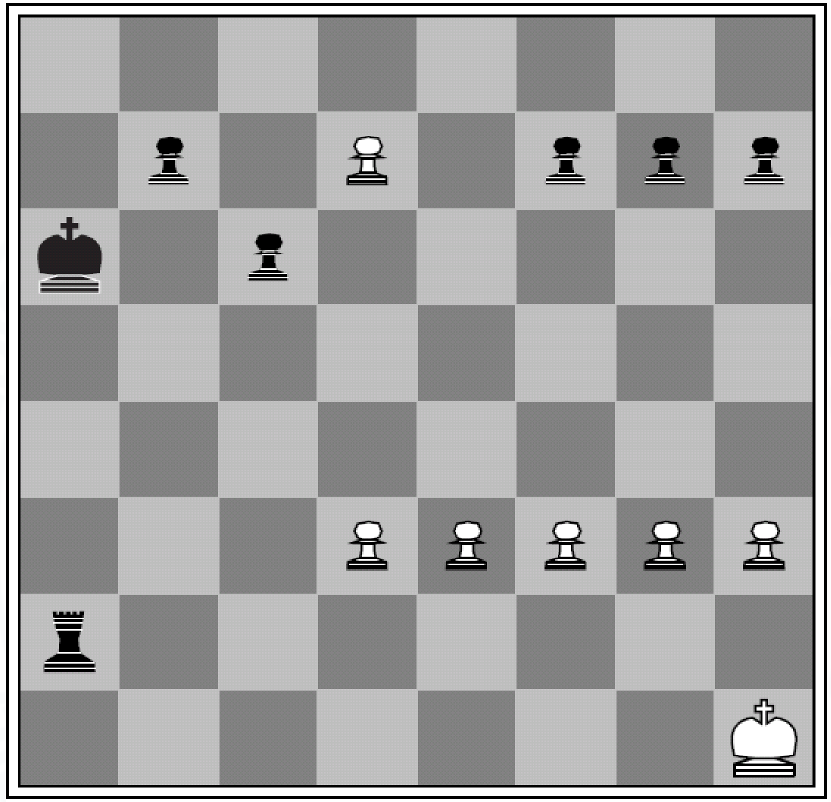


(b) White to move

If B had looked forward one more ply then it would have seen that the Black Queen is in threat: A more sophisticated cutoff procedure is required.

Do not stop the search if the current state is unstable.

# Horizon Effect



Black to move

Black is in an apparently superior position but White can form a queen through its pawn
Black would tend to 'check' the King to avoid this queening but ultimately the pawn will become a queen
The problem with fixed depth search is that it believes that such a move would prevent queening, but the truth is that the queening move has been pushed over the horizon and cannot be seen

# Evaluation functions

- For chess, typically linear weighted sum of features

  $$Eval(s) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s)$$

- e.g., $w_1 = 9$ with

  $f_1(s)$ = (number of white queens) – (number of black queens), etc.

# Cutting off search

*MinimaxCutoff* is identical to *MinimaxValue* except

1. *Terminal?* is replaced by *Cutoff?*
2. *Utility* is replaced by *Eval*

Does it work in practice?

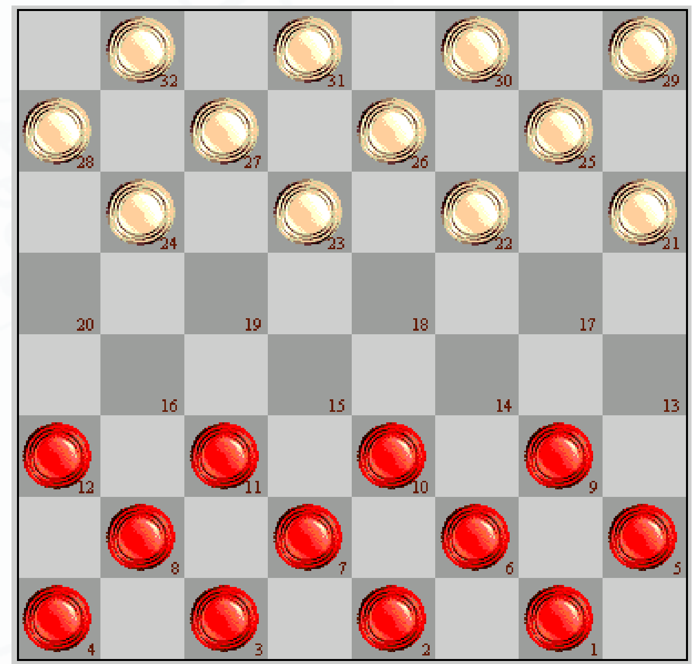
$b^m = 10^6$, $b=35 \rightarrow m=4$

4-ply lookahead is a hopeless chess player!

- 4-ply ≈ human novice
- 8-ply ≈ typical PC, human master
- 12-ply ≈ Deep Blue, Kasparov

# Deterministic games in practice: Checkers

- Checkers was solved on April 29, 2007 by the team of Jonathan Schaeffer, known for *Chinook*, the *"World Man-Machine Checkers Champion"*
- From the standard starting position, both players can guarantee a draw with perfect play
- Checkers is the largest game that has been solved to date, with a search space of $5 \times 10^{20}$
- The number of calculations involved were $10^{14}$ and were done over a period of 18 years. The process involved from 200 desktop computers at its peak down to around 50

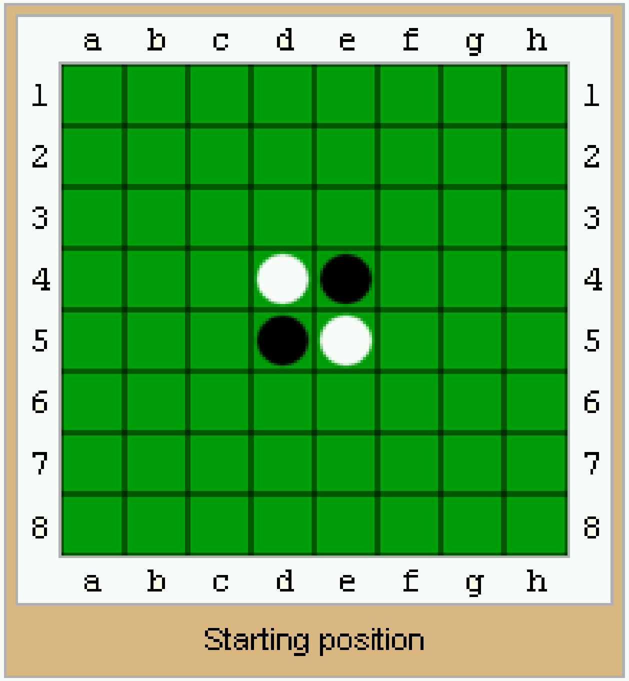


# Deterministic games in practice: Chess

- Deep Blue defeated human world champion Garry Kasparov in a six-game match in 1997. Deep Blue searches 200 million positions per second, uses very sophisticated evaluation, and undisclosed methods for extending some lines of search up to 40 ply.
- Solved by retrograde computer analysis for all 3- to 6-piece, and some 7-piece endgames, counting the two kings as pieces. It is solved for all 3–3 and 4–2 endgames with and without pawns, where 5-1 endgames are assumed to be won with some trivial exceptions
- The full game has 32 pieces. Chess on a 3x3 board is strongly solved by Kirill Kryukov (2004)

# Deterministic games in practice: Othello

- Human champions refuse to compete against computers, who are too good!



Starting position

# Deterministic games in practice: Go

- Human champions refuse to compete against computers, who are too bad (ca. 2007)
- In go, *b > 300*, so most programs use pattern knowledge bases to suggest plausible moves

- AlphaGo
  - Beat the human world champion
  - Uses Deep Neural Networks to predict the utility of a move

Must watch:
http://www.nature.com/news/google-ai-algorithm-masters-ancient-game-of-go-1.19234

# Deterministic games in practice: Robocup

- Initiated in 1993
- Autonomous robots play soccer!
- Official Goal Statement
  - *By mid-21st century, a team of fully autonomous humanoid robot soccer players shall win the soccer game, complying with the official rule of the FIFA, against the winner of the most recent World Cup*

# End of Lecture

Humans Are the World's Best Pattern-Recognition Machines, But for How Long?

http://bigthink.com/endless-innovation/humans-are-the-worlds-best-pattern-recognition-machines-but-for-how-long