CSE 604

Artificial Intelligence

Chapter 5: Adversarial Search

Adapted from slides available in Russell & Norvig's textbook webpage

Dr. Ahmedul Kabir

IIT University of Dhaka

Outline

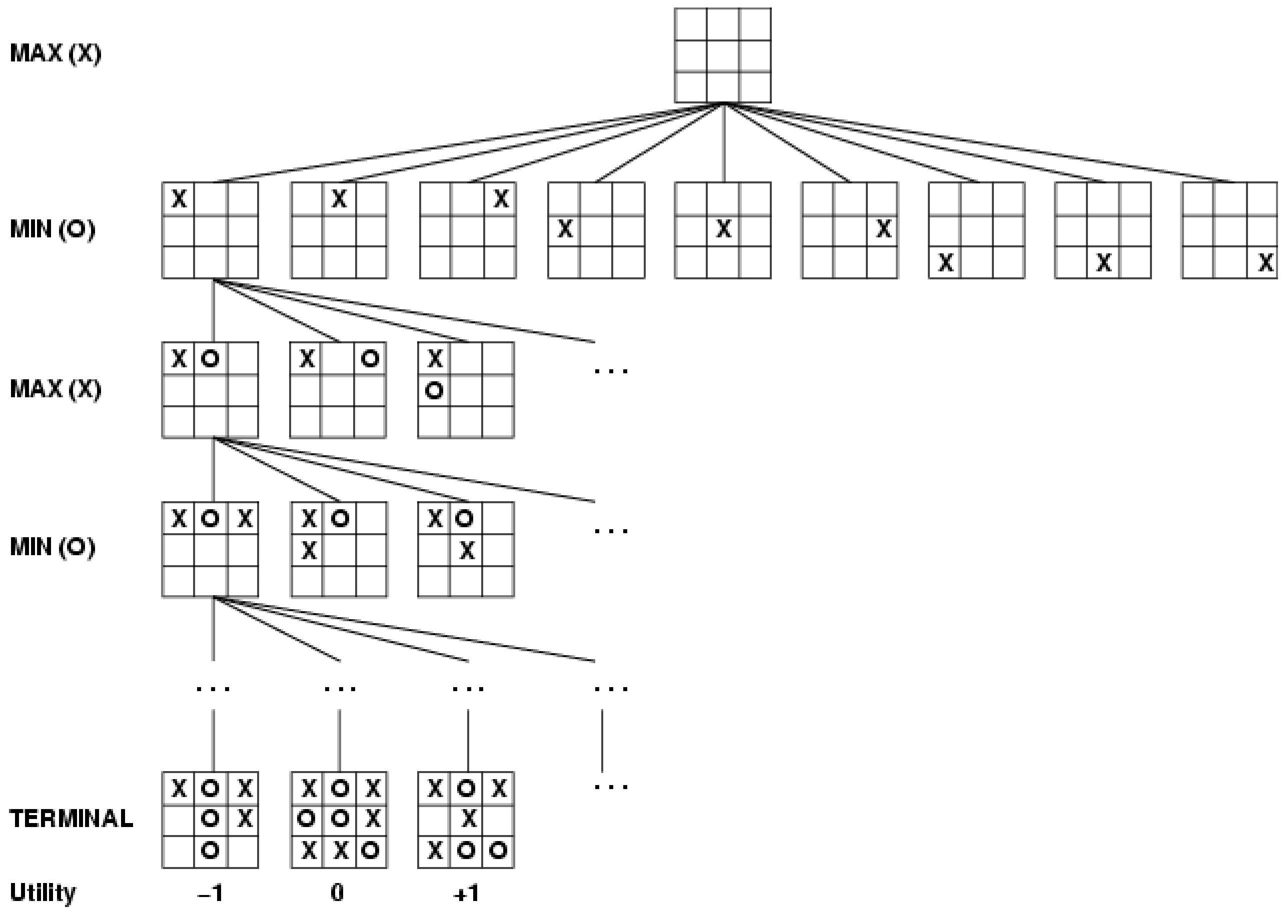
- Optimal decisions
- α-β pruning
- Imperfect, real-time decisions

Games vs. search problems

- "Unpredictable" opponent → specifying a move for every possible opponent reply

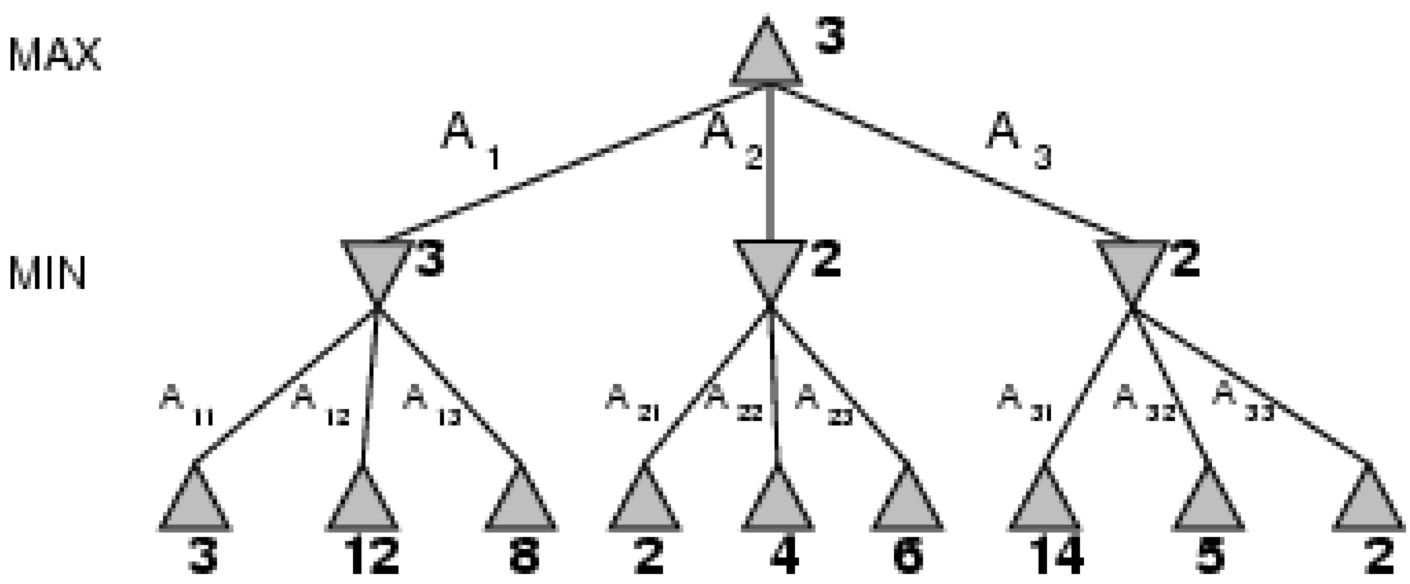
- Time limits → unlikely to find goal, must approximate

Game tree (2-player, deterministic, turns)



Minimax

- Perfect play for deterministic games
- Idea: choose move to position with highest minimax value
  = best achievable payoff against best play
- E.g., 2-ply game:



Minimax algorithm

function MINIMAX-DECISION(*state*) **returns** *an action*

  $v \leftarrow$ MAX-VALUE(*state*)
  return the *action* in SUCCESSORS(*state*) with value v

---

function MAX-VALUE(*state*) **returns** *a utility value*

  if TERMINAL-TEST(*state*) **then return** UTILITY(*state*)
  $v \leftarrow -\infty$
  for a, s in SUCCESSORS(*state*) **do**
    $v \leftarrow$ MAX(v, MIN-VALUE(s))
  return v

---

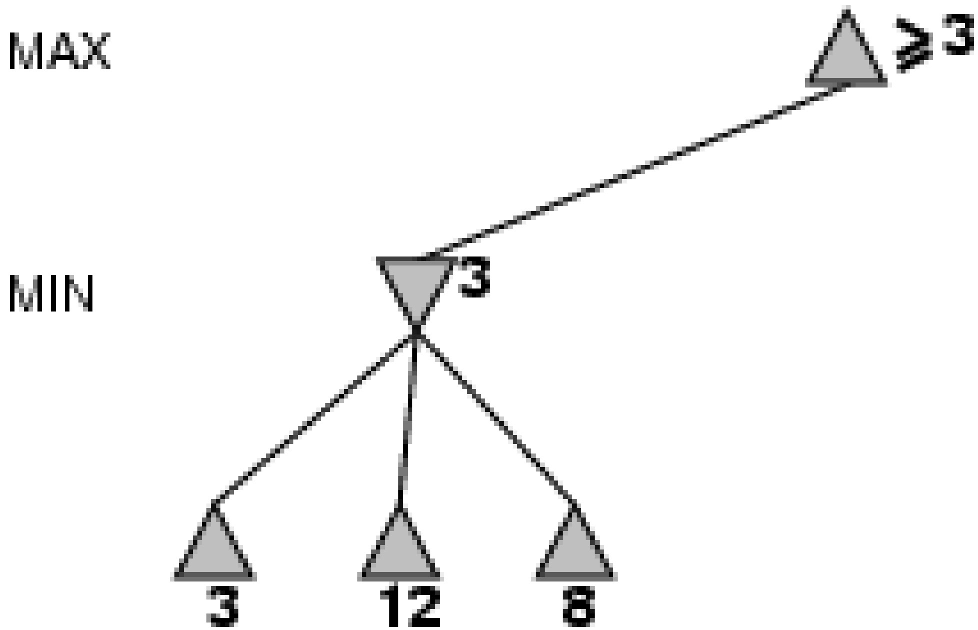
function MIN-VALUE(*state*) **returns** *a utility value*

  if TERMINAL-TEST(*state*) **then return** UTILITY(*state*)
  $v \leftarrow \infty$
  for a, s in SUCCESSORS(*state*) **do**
    $v \leftarrow$ MIN(v, MAX-VALUE(s))
  return v

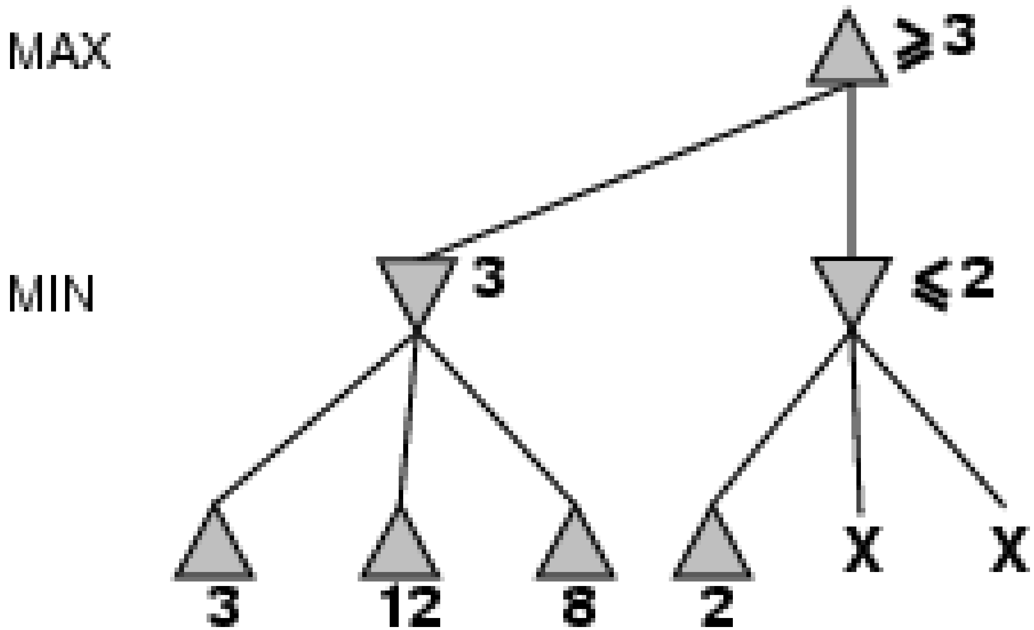
Properties of minimax

- 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

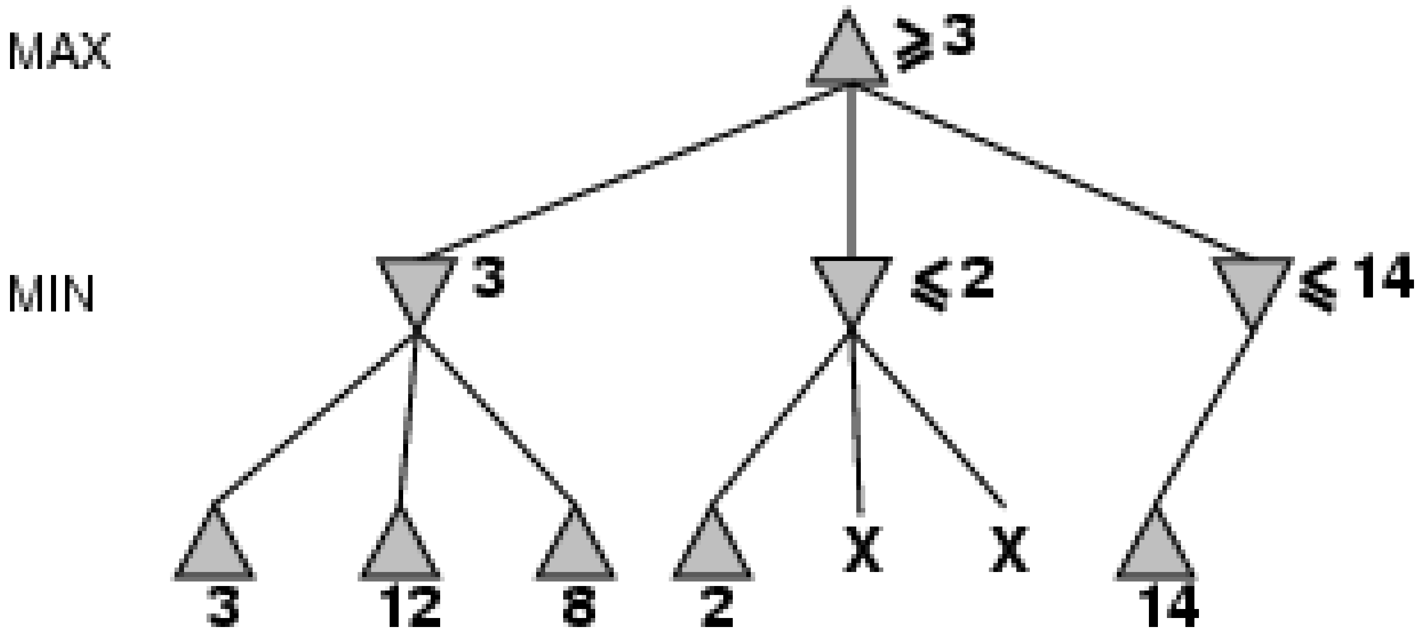
α-β pruning example



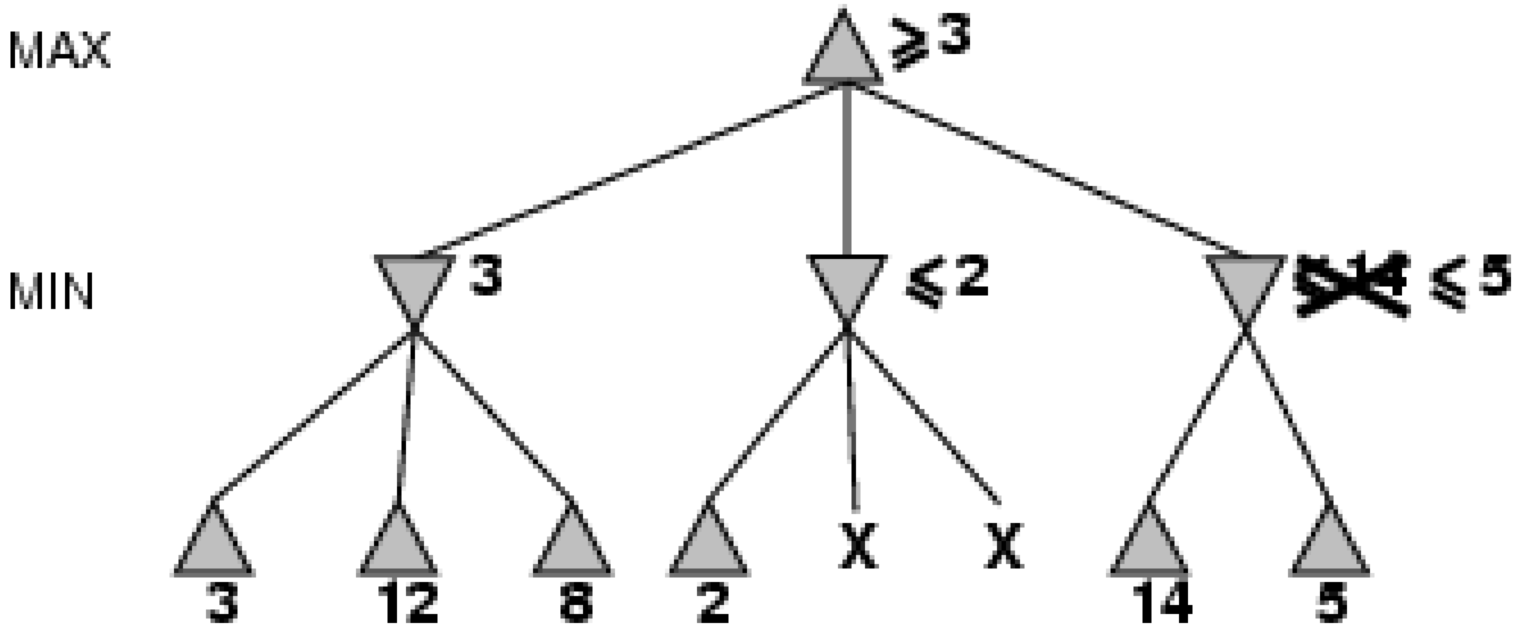
α-β pruning example



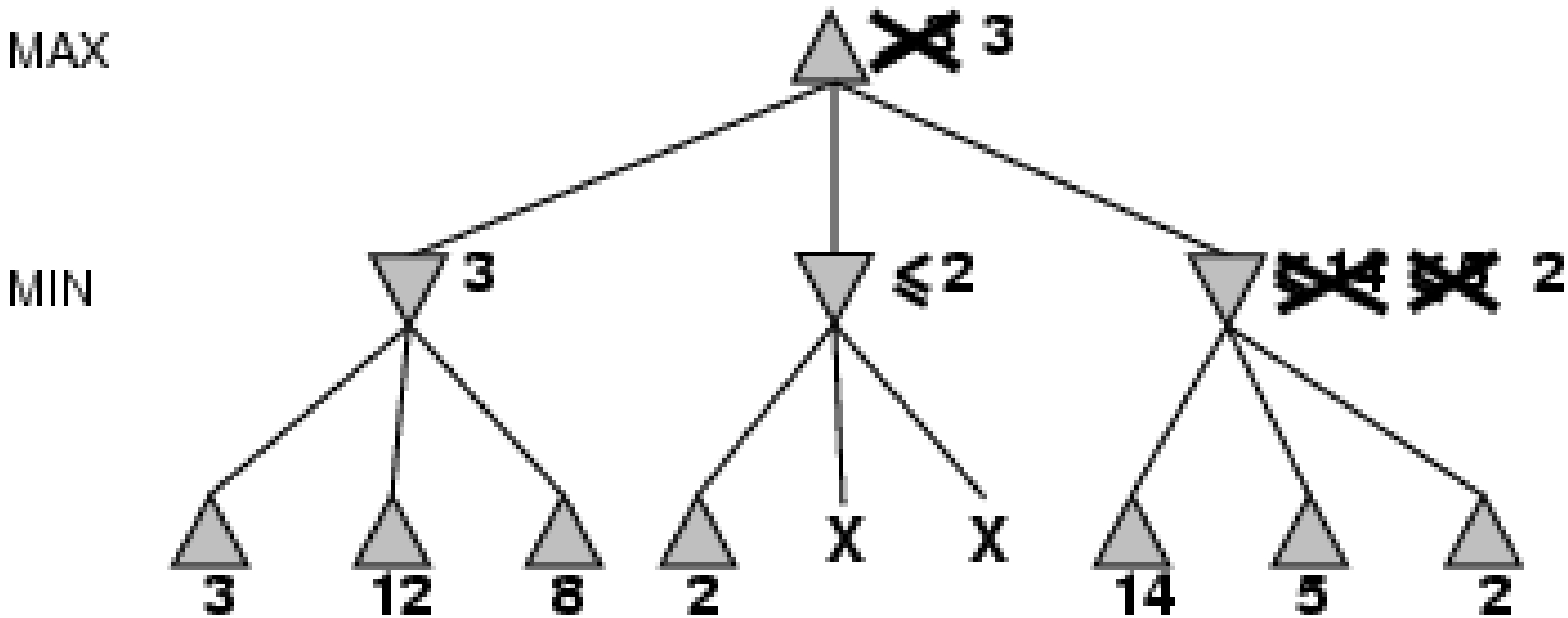
α-β pruning example



α-β pruning example



α-β pruning example



Properties of α-β

- Pruning does not affect final result

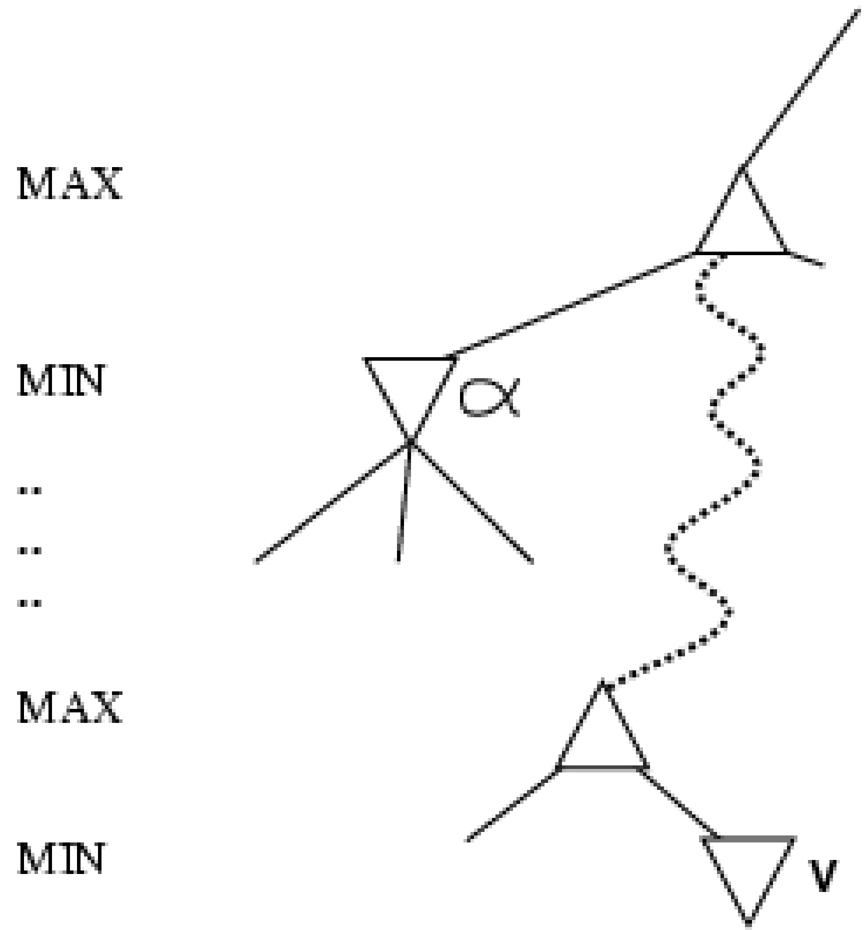
- Good move ordering improves effectiveness of pruning
  - Try the moves that are "likely to be best" first
  - E.g., in chess, try captures, threats, forward moves, backward moves in that order

- With "perfect ordering," time complexity = $O(b^{m/2})$
  → doubles depth of search

- A simple example of the value of reasoning about which computations are relevant (a form of metareasoning)

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($state, \alpha, \beta$) **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*)
  $v \leftarrow +\infty$
  for a, s in SUCCESSORS(*state*) **do**
    $v \leftarrow$ MIN(v, MAX-VALUE(s, α, β))
    if $v \leq \alpha$ **then return** v
    $\beta \leftarrow$ MIN(β, v)
  return v

Imperfect Real Time Decisions

Even with alpha-beta pruning, it is infeasible to grow the whole game tree!

Standard approach:

- evaluation function
  = estimated desirability of position
- cut off search
  e.g., depth limit or iterative deepening
- forward pruning
  e.g., Beam search

Evaluation functions

- For chess, typically linear weighted sum of features

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

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

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

  $w_2 = 5$ with

  $f_2(s)$ = (number of white rooks) – (number of black rooks),

etc.

Cutting off search

- We can use a modified algorithm *MinimaxCutoff*
- *MinimaxCutoff* is identical to *MinimaxValue* except
  - *Terminal?* is replaced by *Cutoff?*
  - *Utility* is replaced by *Eval*

- Does it work in practice?
  - Suppose we have 100 secs, explore 10^4 nodes/sec
    → 10^6 nodes per move

    $b^m = 10^6$, $b=35$ → $m=4$

- 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**: Chinook ended 40-year-reign of human world champion Marion Tinsley in 1994.
- **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.
- **Othello**: human champions refuse to compete against computers, who are too good.
- **Go**: Until recently, human champions refused to compete against computers, who were too bad (in Go, $b > 300$).
  - But in 2016, Google's AlphaGo defeated human world champion Lee Sedol.
  - In 2017, AlphaGo Zero defeated the previous version of AlphaGo 100-0