University of BRISTOL | Department of Computer Science

PROGRAMMING and ALGORITHMS II

INTRODUCTION TO

ADVERSARIAL SEARCH

MINIMAX AND ALPHA-BETA REDUCTION

Dr Tilo Burghardt

Unit Code COMS10001

Algorithm Design Paradigms

DIVIDE & CONQUER

Break down a problem into **independent** sub-problems of related type, solve them separately and combine the solutions.

GREEDY APPROACH

Use a sequence of **locally optimal** decisions incrementally to build up a solution.

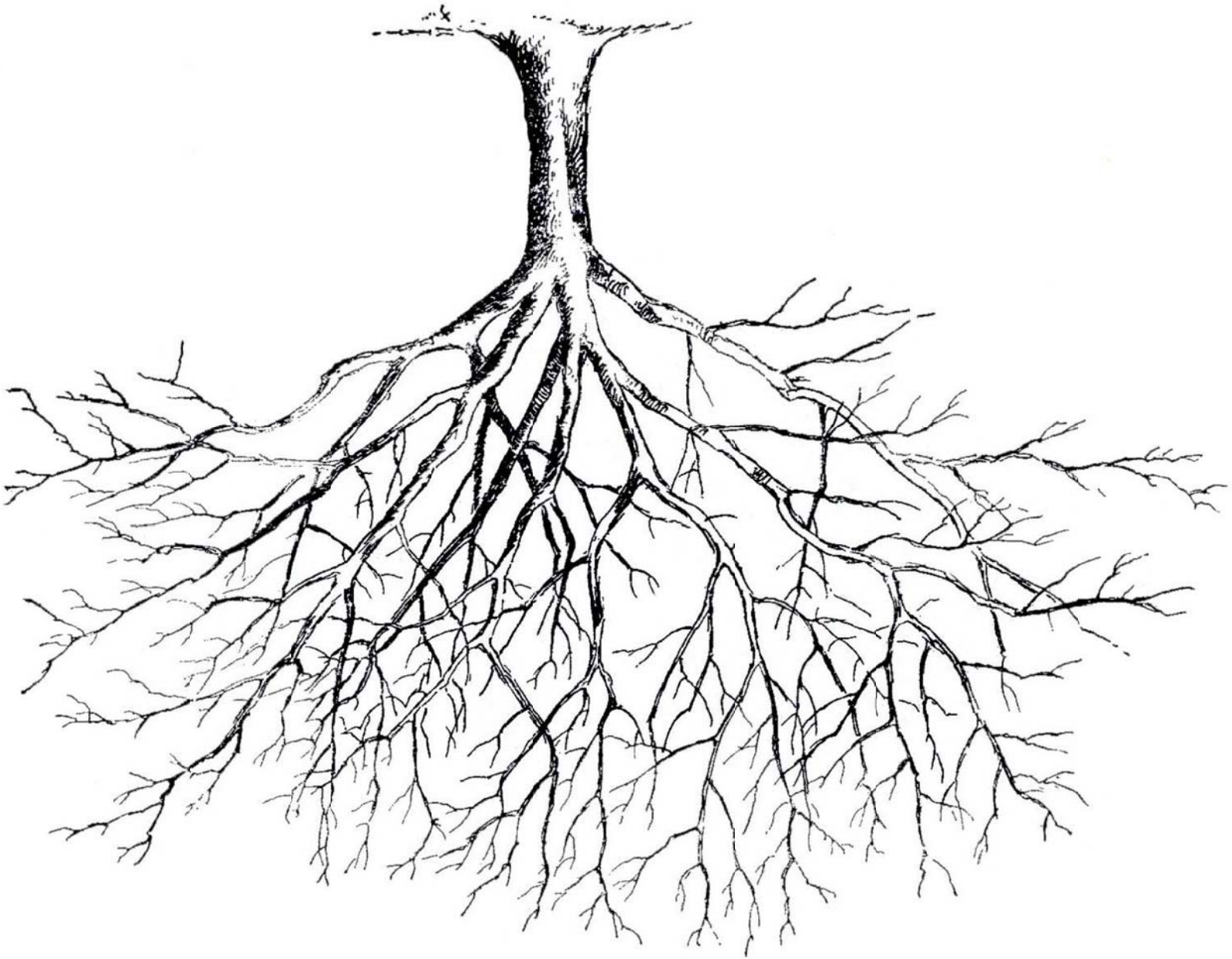
DYNAMIC PROGRAMMING

Break down a problem into **overlapping** sub-problems of related type, build up solutions from larger and larger sub-solutions.

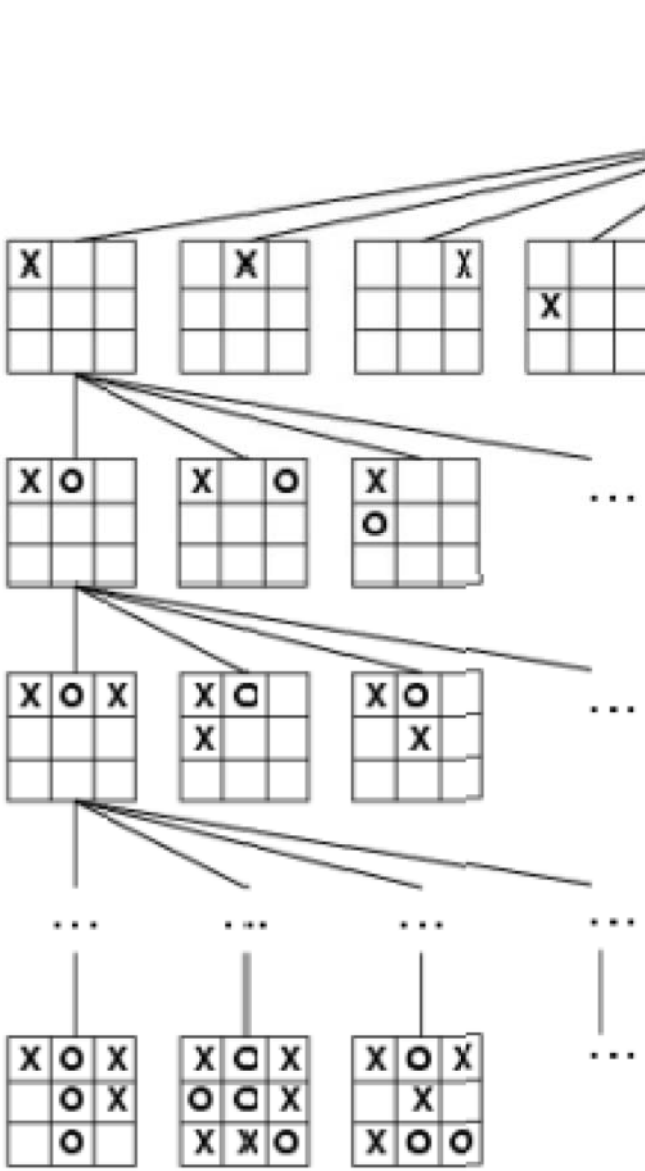
Deep Blue

- In 1996 Deep Blue became the first machine to win a chess game against a reigning world champion (Garry Kasparov) under regular time controls
- 30 nodes running at 120 MHz
- program written in C exploring a 'game tree' at 200 million positions per second

Concept of a ‘Game Tree’



Semantics of a ‘Game Tree’

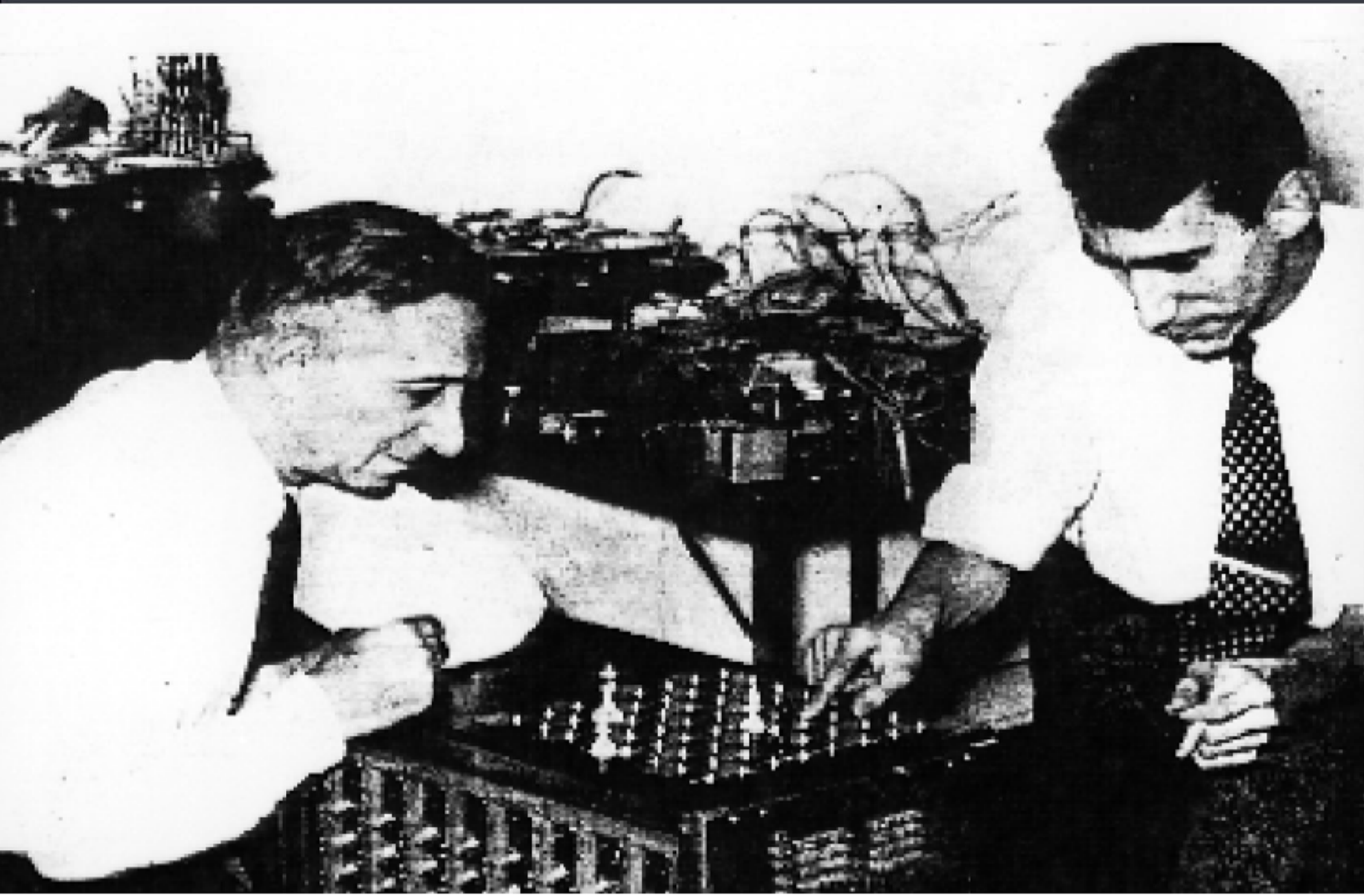


- Nodes are states
- Edges are decisions
- Levels are called “plys”

We will look at deterministic, turn-taking games of perfect information first

Shannon's Chess Machine

University of BRISTOL | Department of Computer Science



Origins of MiniMax Algorithm

- In 1950 Shannon published paper: *Programming a Computer for Playing Chess*.
- General Idea: Computer simulates sequences of the enemy as well as own moves.
- The process for having the computer decide on which move to make is guided by trying to **Min**imize the **max**imum possible loss, or Maximize the minimum win…

→ Find winning sequence where enemy plays optimally, e.g. maximizing win utility which is + *1* for win, *0* for tie, and - *1* for loss

Game Tree of OXO

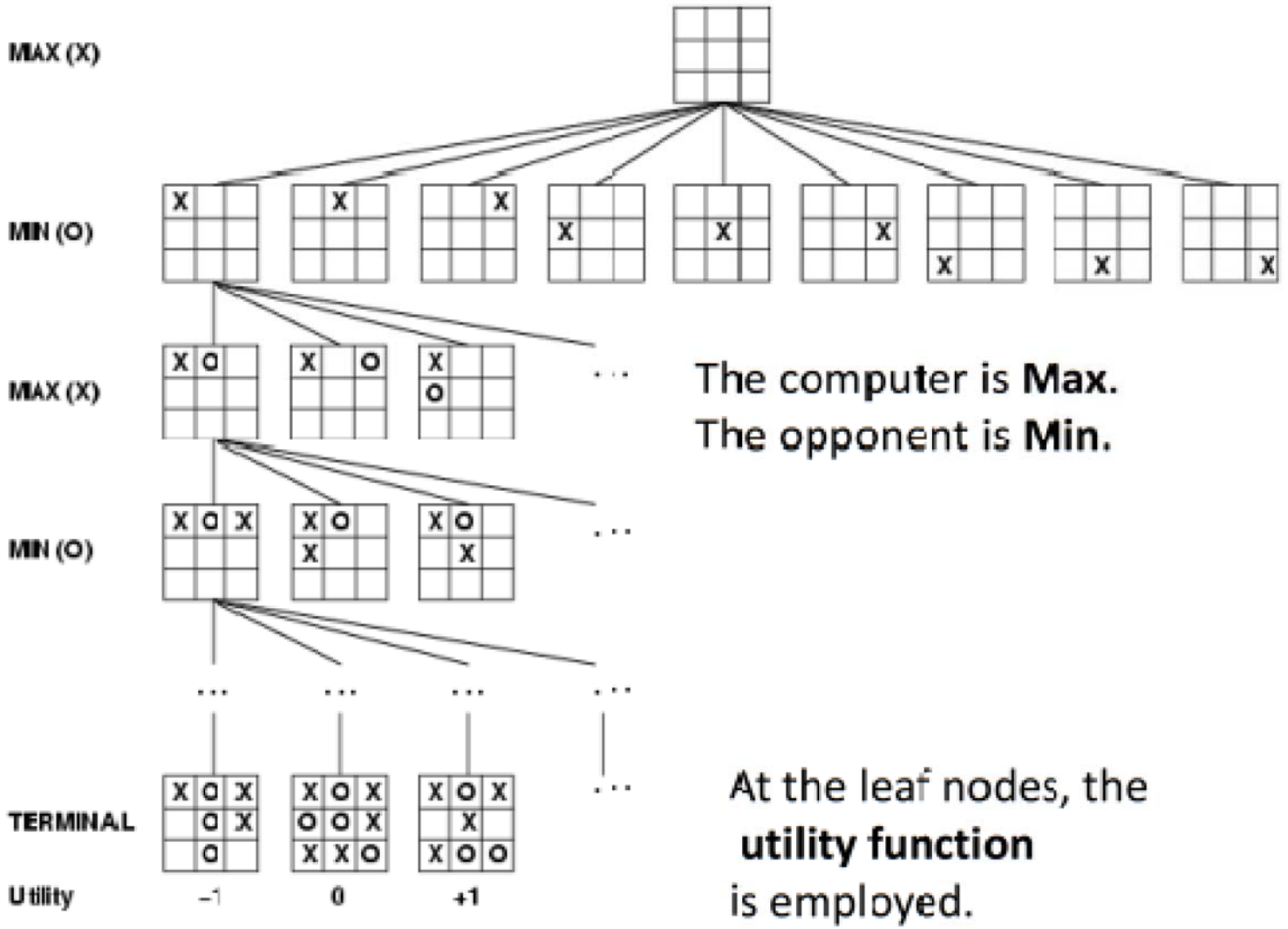
computer's turn

opponent's turn

computer's turn

opponent's turn

leaf nodes are evaluated



The computer is **Max.**
The opponent is **Min.**

At the leaf nodes, the **utility function** is employed.

The MiniMax Algorithm


```
function MINIMAX-DECISION(state) returns an action
    v ← 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 ← −∞
    for s in SUCCESSORS(state) do
        v ← MAX(v, MIN-VALUE(s))
    return v
```

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

MiniMax Properties

- Optimality: Yes (against an optimal opponent)
- Completeness: Yes (if tree is finite)
- Time complexity: $O(m^n)$
- Space complexity: $O(mn)$ (depth-first exploration)

Example: Chess … m ≈ 30 options per move, n ≈ 100 moves for "reasonable" games
→ exact solution completely infeasible

Practical Evaluation

- Assume we can evaluate 10,000 boards per second and we have 10 secs → 10^5 nodes per move

Example: Chess.... $m^n = 10^5, m = 30 \rightarrow n \approx 3$

3-ply look-ahead is a truly hopeless chess player!

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

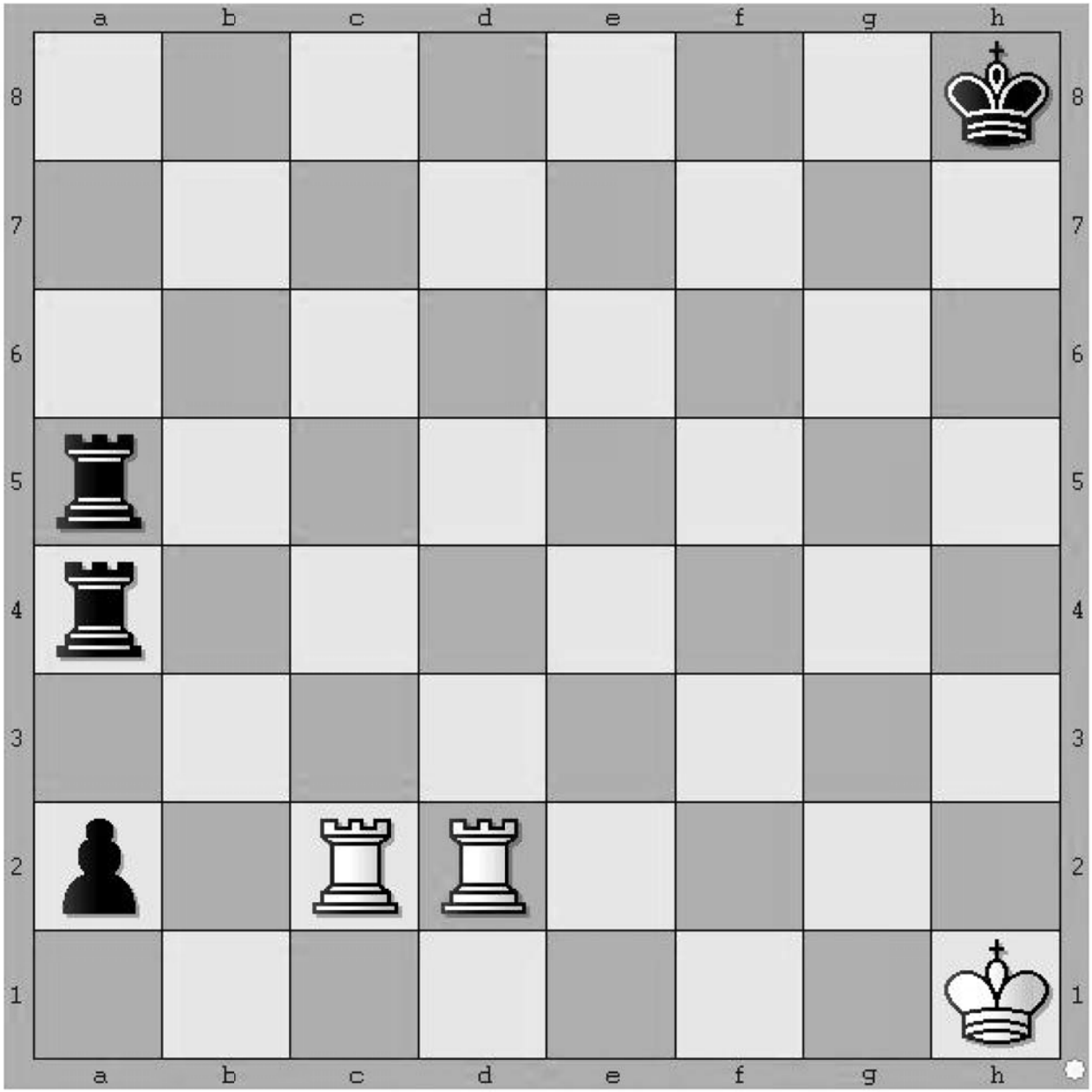
(Remember: A computer which evaluates its own legal moves plus the legal responses to those moves is searching to a depth of 2-ply...)

Shannon's Chess Heuristics

Shannon gave a rough example of an evaluation function for chess in which the value of the black position was subtracted from that of the white position.

- **Piece Value**: was counted according to the relative chess piece value (1 point for a pawn, 3 points for a knight or bishop, 5 points for a rook, and 9 points for a queen).
- **Positional Factors**: subtracting ½ point for each doubled pawns, backward pawn, isolated pawn.
- **Mobility**: adding 0.1 point for each legal move available.
- **Checkmate**: king has artificial value of 200 points

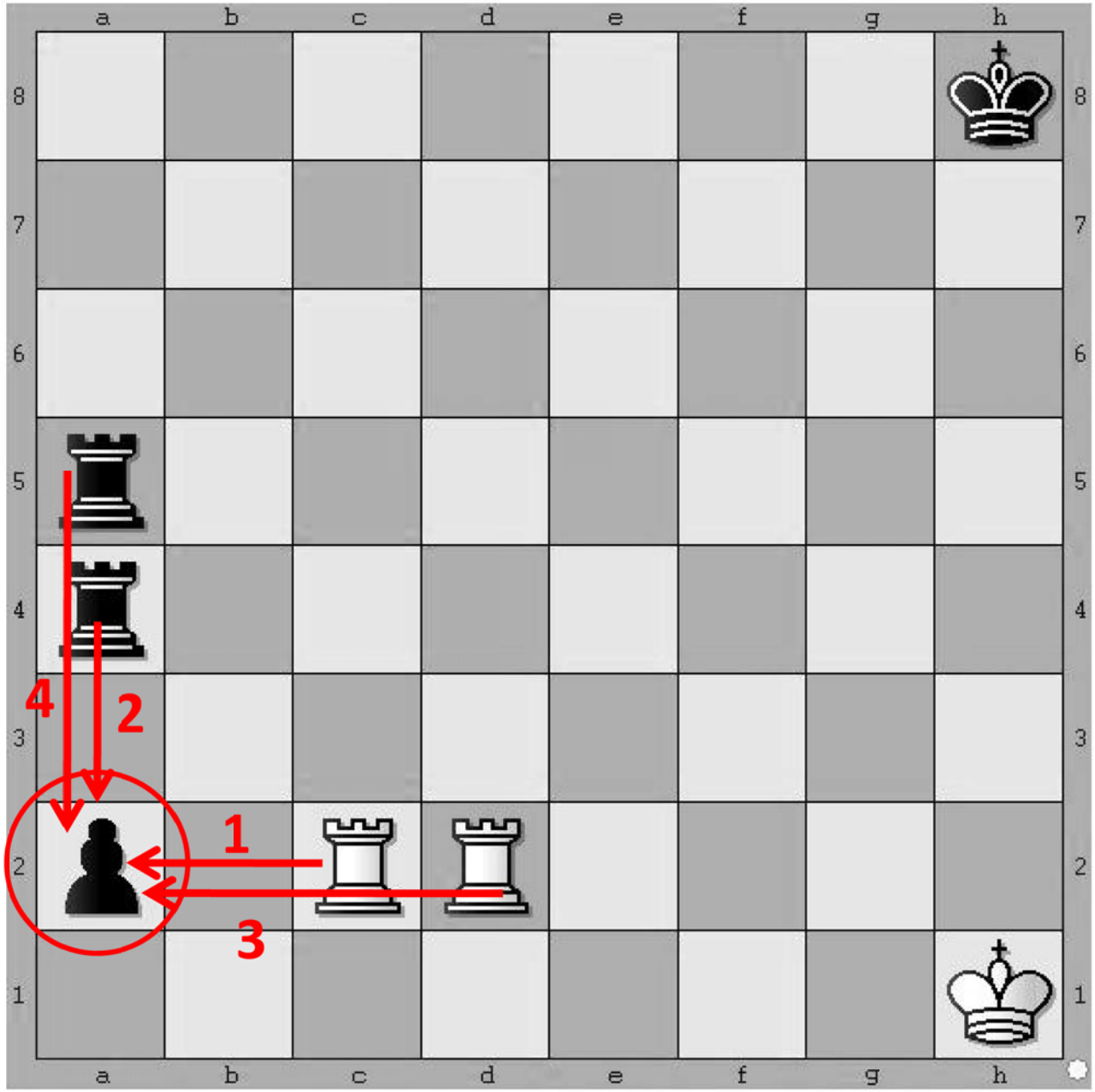
Example: Look-ahead 3



Example by Jonathan Pettersson

Horizon Effect Example

University of BRISTOL | Department of Computer Science



Example by Jonathan Pettersson

Horizon Effect

University of BRISTOL | Department of Computer Science

The Horizon Effect

- Your algorithm searches to depth *n*
- What happens if:
  - Evaluation(s) at depth n is very positive
  - Evaluation(s) at depth n+1 is very negative
- Or:
  - Evaluation(s) at depth n is very negative
  - Evaluation(s) at depth n+1 is very positive

Will this ever happen in practice?

Quiescense Search

- Often it is useful to look deeper into game tree

→ peak past the horizon…

- What nodes to explore?
- Human players have some intuition about move quality
  - "Interesting" vs "boring"
  - "Promising" vs "dead end"
- Expand horizon for potential high impact moves
- Quiescence Search finds strong changes OR consistency along move sequences

Quiescense Search Explosion

- quiescent search does not have a set depth
- a bad ordering of the moves may result in extremely deep lines of search
  → this is called a *quiescent search explosion*
- ordering the search space can provide a limiting criterion, e.g. MVV/LVA (Most Valuable Victim/Least Valuable Attacker)
- this way we are not calculating obvious losses early in the search
- However, horizon effect is still present: we can not know if captured pieces are protected. (A queen capturing an unprotected pawn may be a good move, but a majority of the time the pieces will be protected)

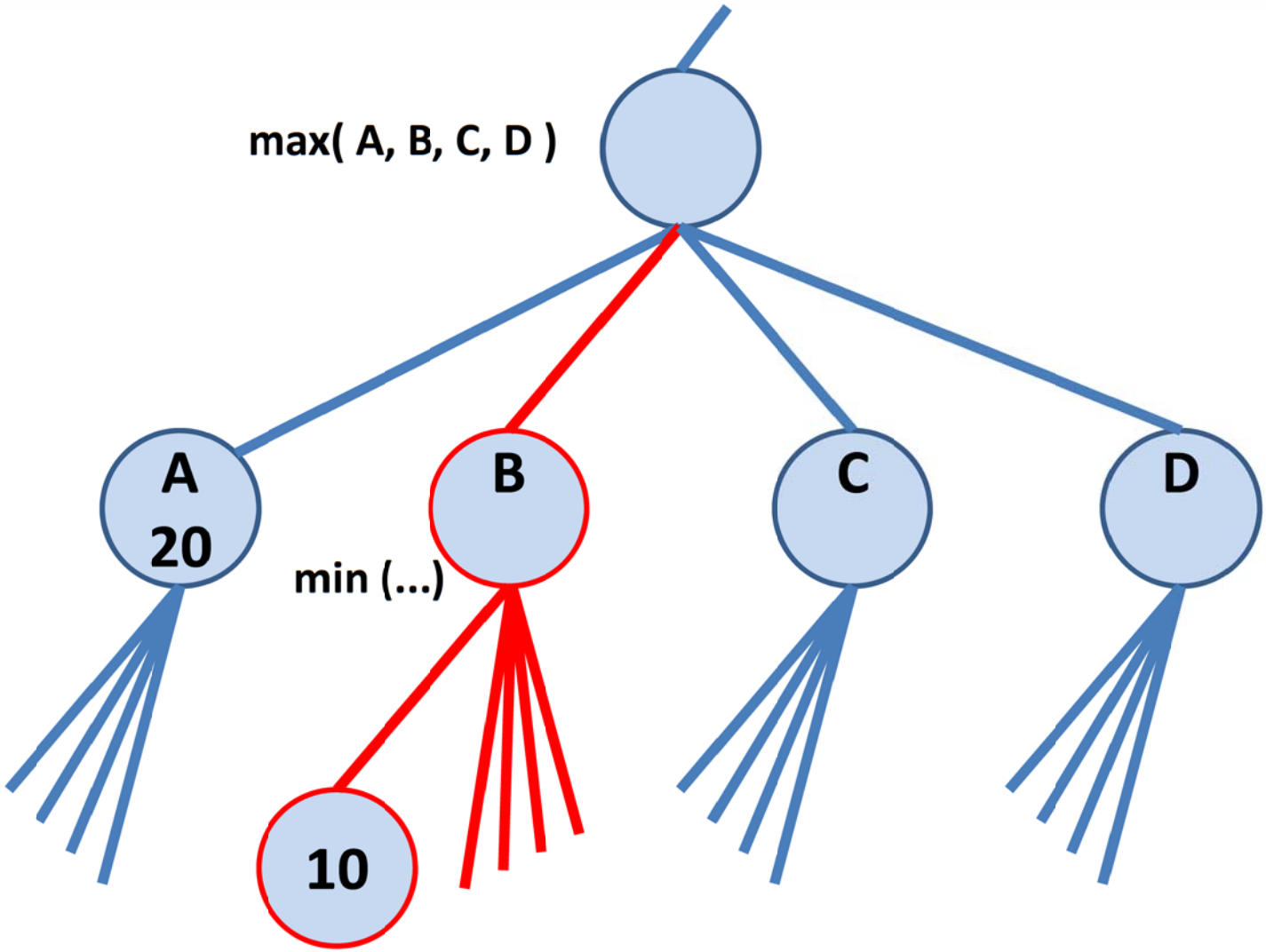
Pre-calculation Strategies

- **Opening Books:** pre-compute opening move sequences and store them for direct access (i.e. pre-analyse root region of game tree)
- **Endgame Databases:** store endgame patterns and exhaustively pre-compute endgame (i.e. pre-analyse leaf regions of game tree)
- **Killer Move Analysis:** store and re-identify configurations likely to lead to endgame (i.e. identify critical/forced paths to end game)
- **Transposition (Dynamic Programming) Tables**: different move sequences may result in identical boards, computing solutions once and storing analysis results can avoid double exploration

Pruning Game Trees

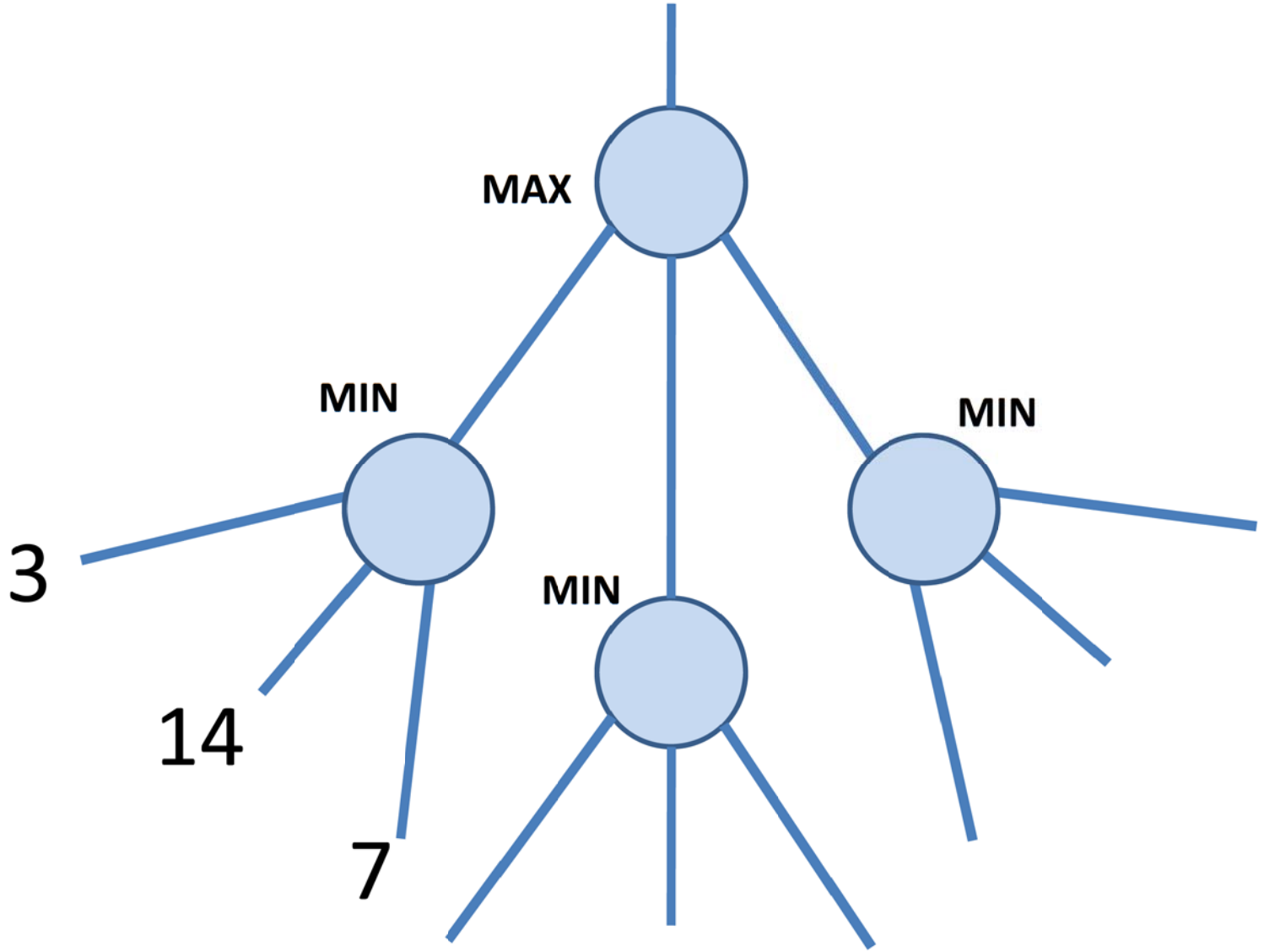
University of BRISTOL | Department of Computer Science

Idea of Game Tree Pruning

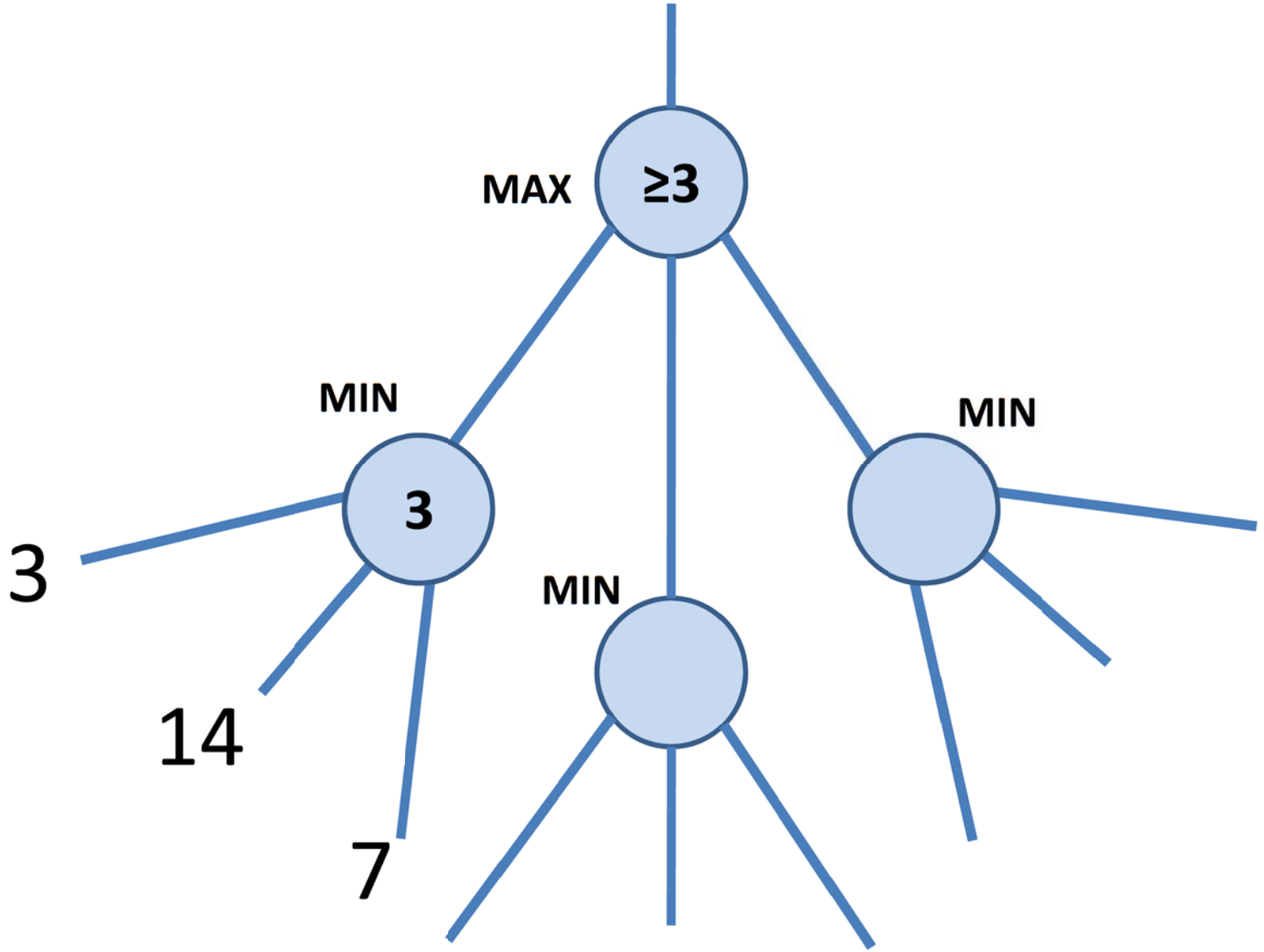


Node B and its entire subtree have no influence on the value of parent node, since max(20,...)>=20 and B<=10...

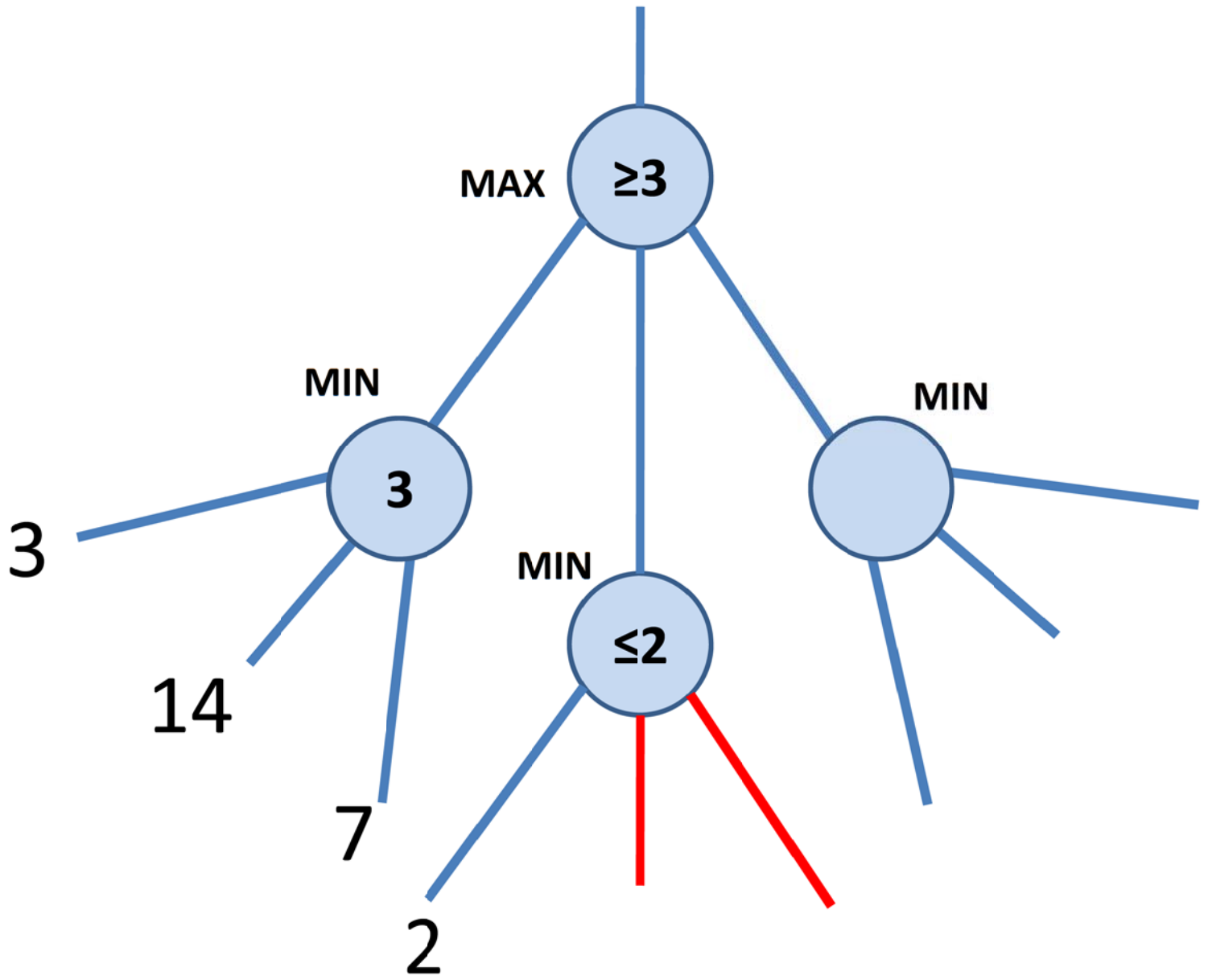
Alpha-Beta Pruning Example



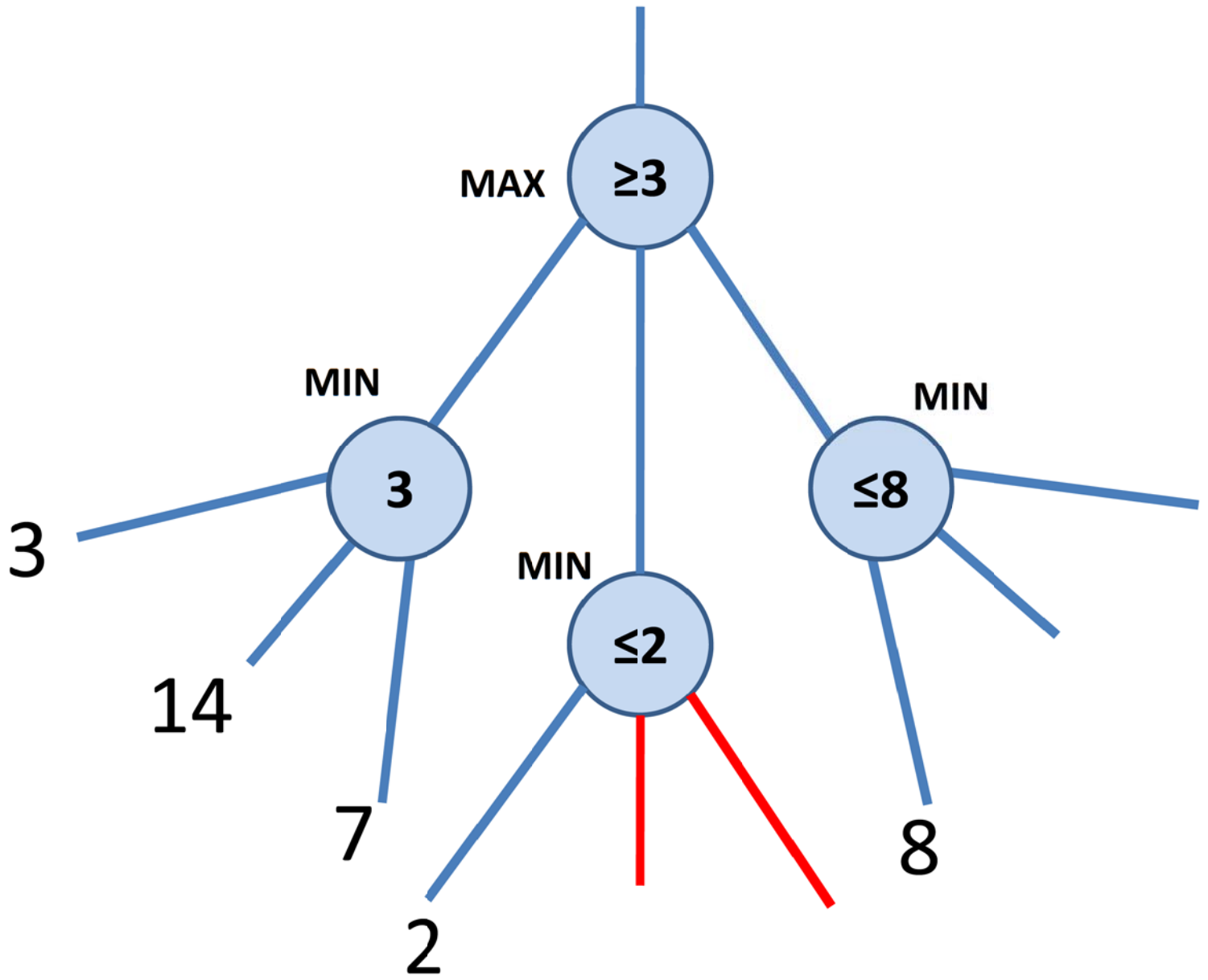
Alpha-Beta Pruning Example



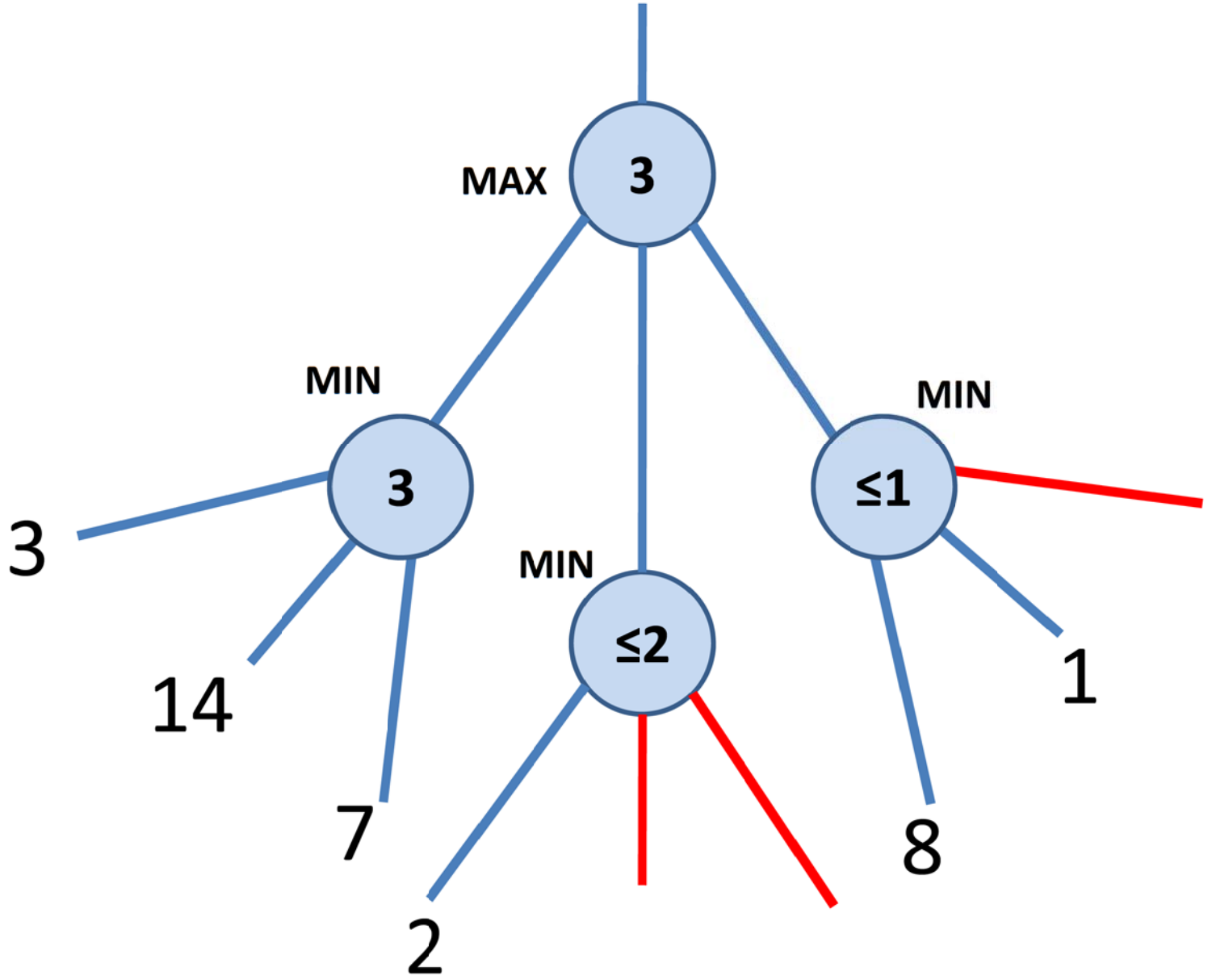
Alpha-Beta Pruning Example



Alpha-Beta Pruning Example



Alpha-Beta Pruning Example



Alpha-Beta Properties

- Alpha-Beta is computes the same value for the root node as computed by the MiniMax algorithm
- **Worst Case:** Alpha-Beta does NO pruning, examining m^n leaf nodes, where nodes have *m* children and a *n*-ply search is performed
- **Best Case:** Alpha-Beta will examine only $2m^{n/2}$ leaf nodes. Ergo, the search can now be twice as deep as MiniMax…
- If possible, consider ordering of leaf nodes to optimise pruning performance...

Alpha-Beta Algorithm

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

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

---

function MAX-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*

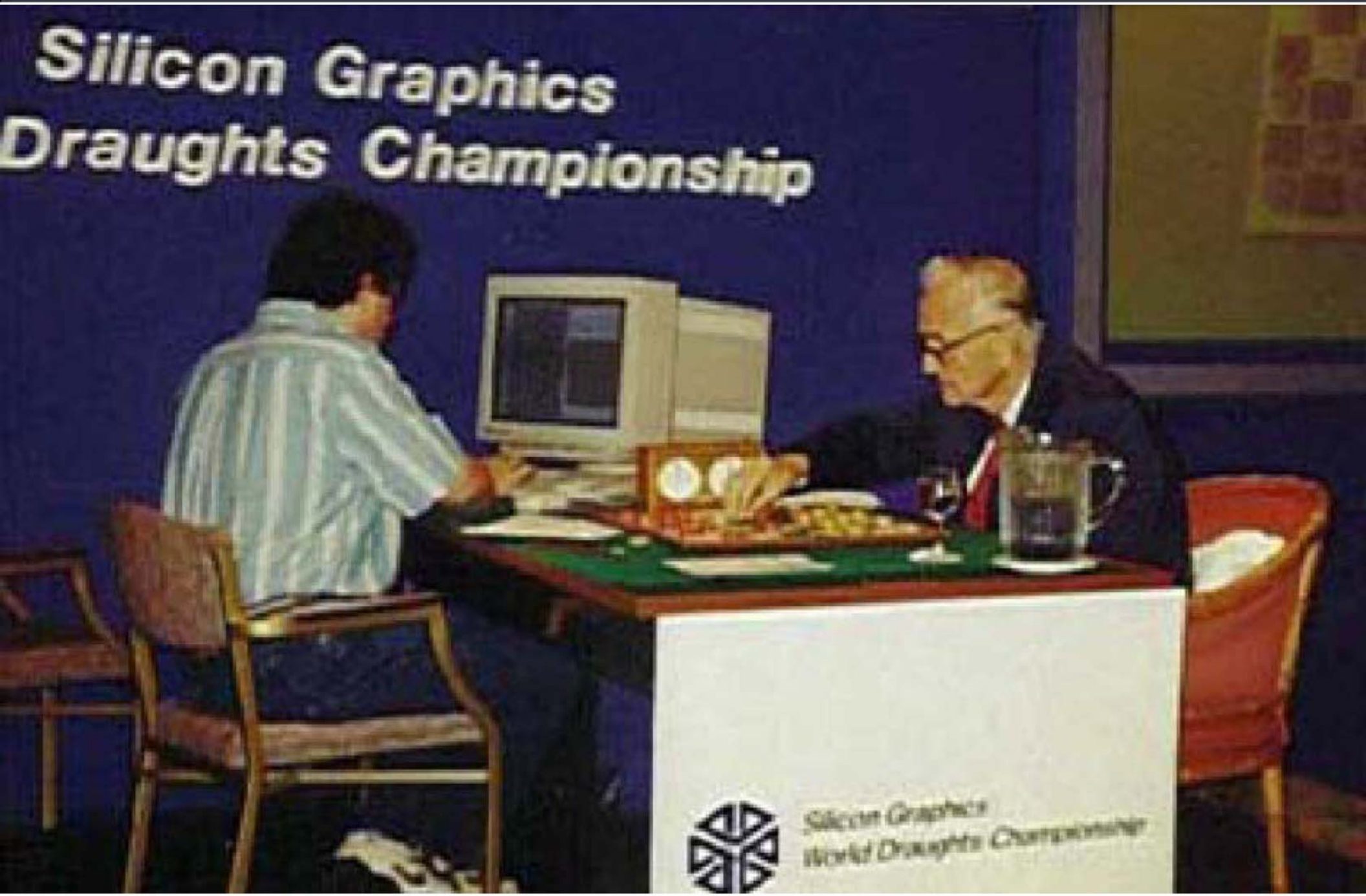
  if TERMINAL-TEST(*state*) **then return** UTILITY(*state*)
  $v \leftarrow -\infty$
  for a, s in SUCCESSORS(*state*) **do**
    $v \leftarrow$ MAX(v, MIN-VALUE(s, α, β))
    if $v \geq \beta$ **then return** v
    $\alpha \leftarrow$ MAX(α, v)
  return v

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($s\ \alpha, \beta$))
    if $v \leq \alpha$ **then return** v
    $\beta \leftarrow$ MIN(β, v)
  return v

Deep Blue vs Kasparov

Draughts Championship

University of BRISTOL | Department of Computer Science



GO is still Human-reigned

University of BRISTOL | Department of Computer Science

Summary

- Game playing can be effectively modeled as a **Search Problem**
- **Game Trees** represent alternate computer/opponent moves
- Evaluation functions estimate the quality of a given board configuration
- **The MiniMax Algorithm** is a procedure which chooses moves by assuming that the opponent will always choose the move which is best for them
- **Alpha-Beta Pruning** is a procedure which can discard large parts of the search tree and, thus, allow search to go deeper