Adversarial Search

Chapter 5

Outline

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

Games vs. search problems

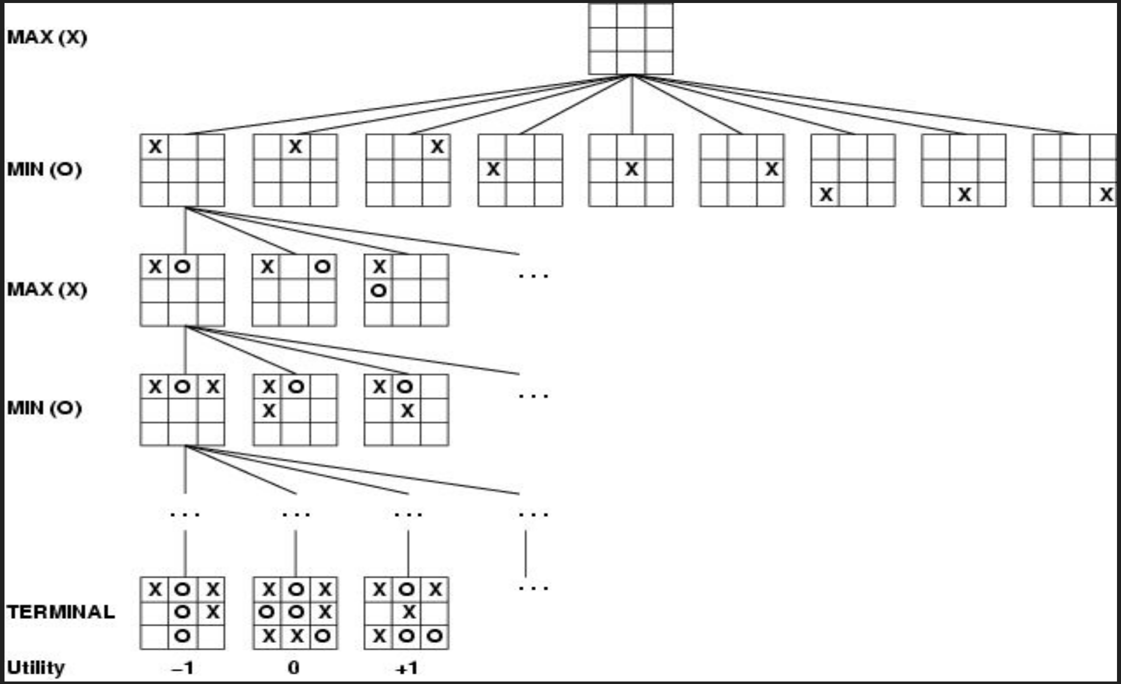
- "Unpredictable" opponent specifying a move for every possible opponent reply
  prisoner dilemma

A\B	betray	persist
betray	0\0	1\-1
persist	-1\1	2\2

- Time limits (35^100) unlikely to find goal, must approximate

Game tree (2-player, deterministic, turns)

tic-tac-toe



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

---

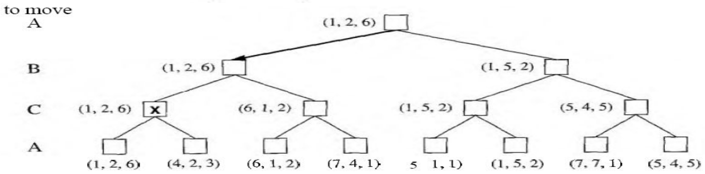
function MAX-VALUE(*state*) **returns** *a utility value*

  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

OPTIMAL STRATEGIES

-Find the contingent *strategy* for MAX assuming an infallible MIN opponent.

-**Assumption**: Both players play optimally !!

Given a game tree, the optimal strategy can be determined by using the minimax value of each node:

-**MINIMAX-VALUE(*n*)=**

UTILITY(*n*) If *n* is a terminal

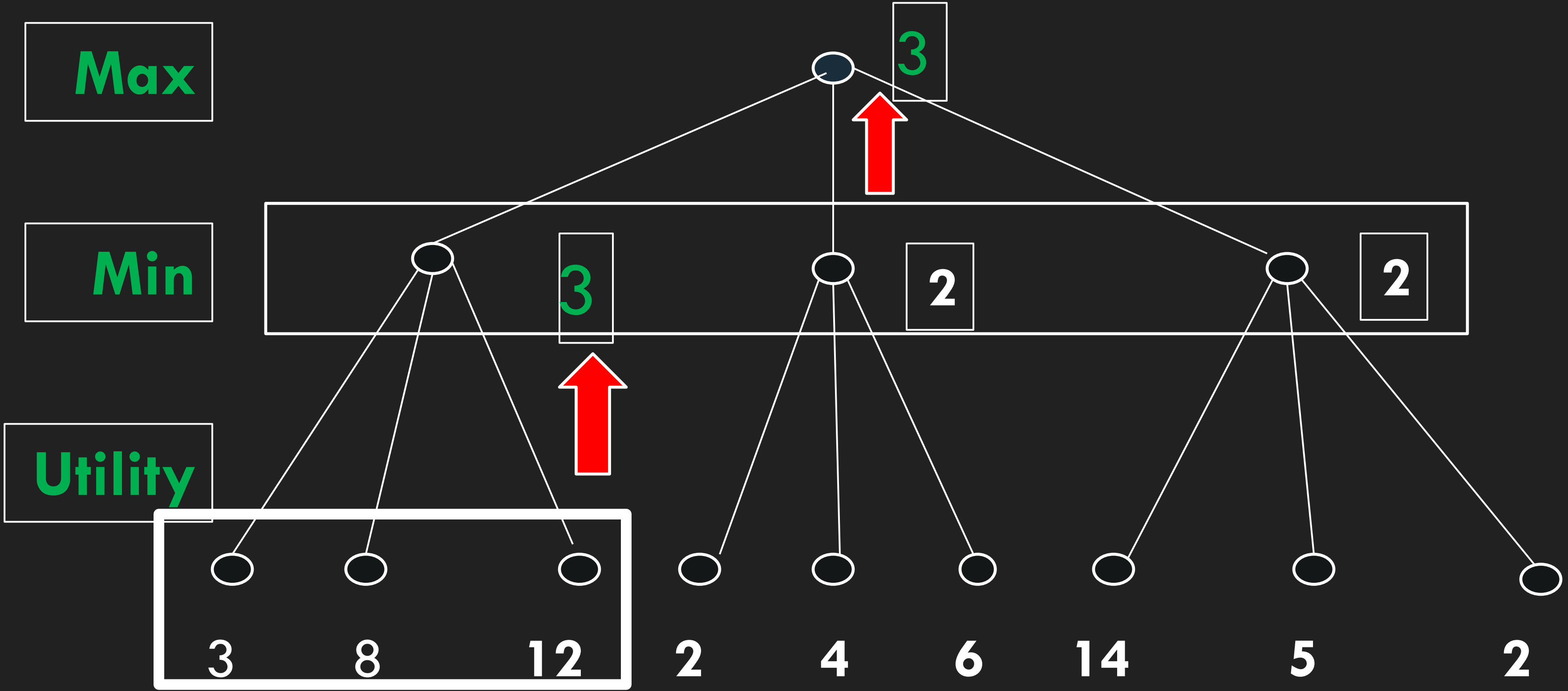
$\max_{s \in successors(n)}$ MINIMAX-VALUE(*s*) If *n* is a max then

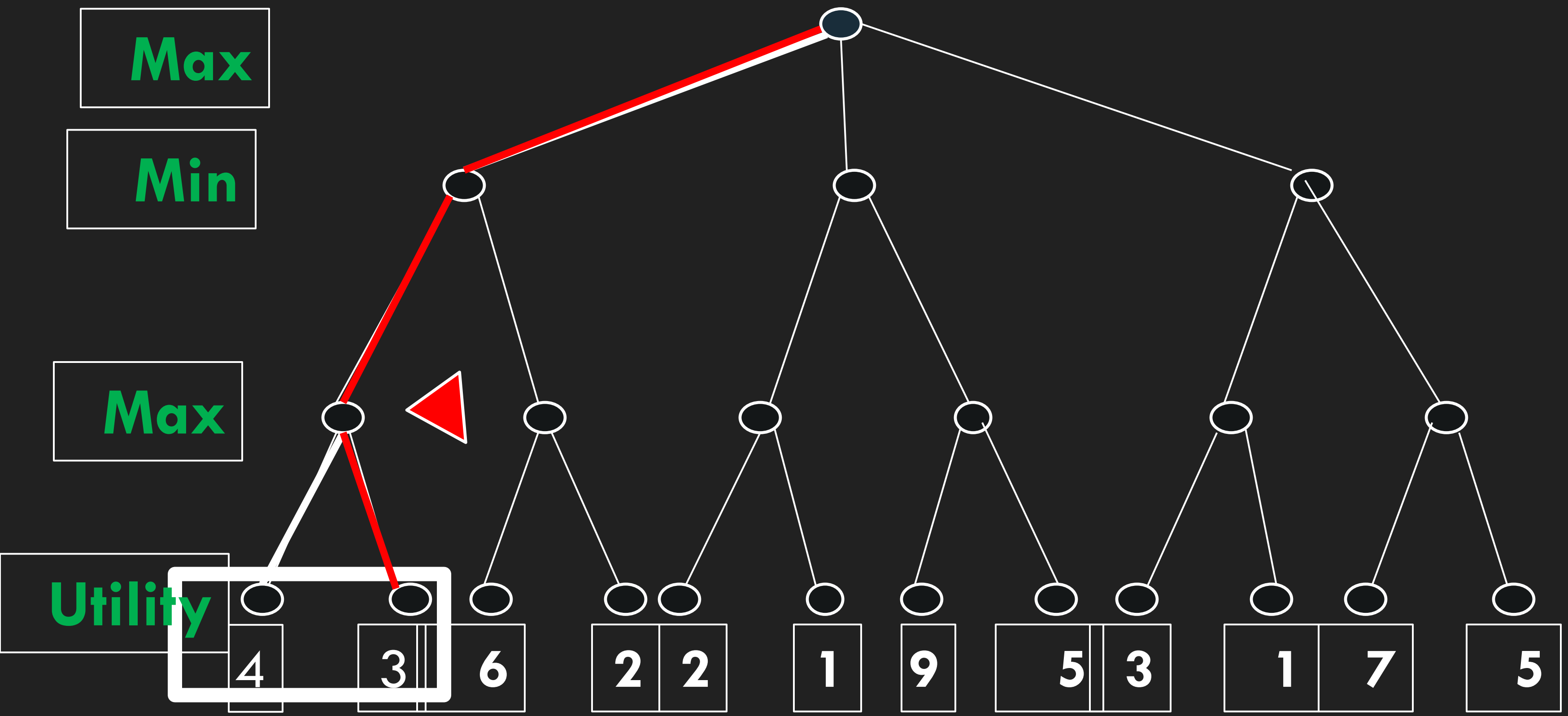
$\min_{s \in successors(n)}$ MINIMAX-VALUE(*s*) If *n* is a min node

MINIMAX ALGORITHM

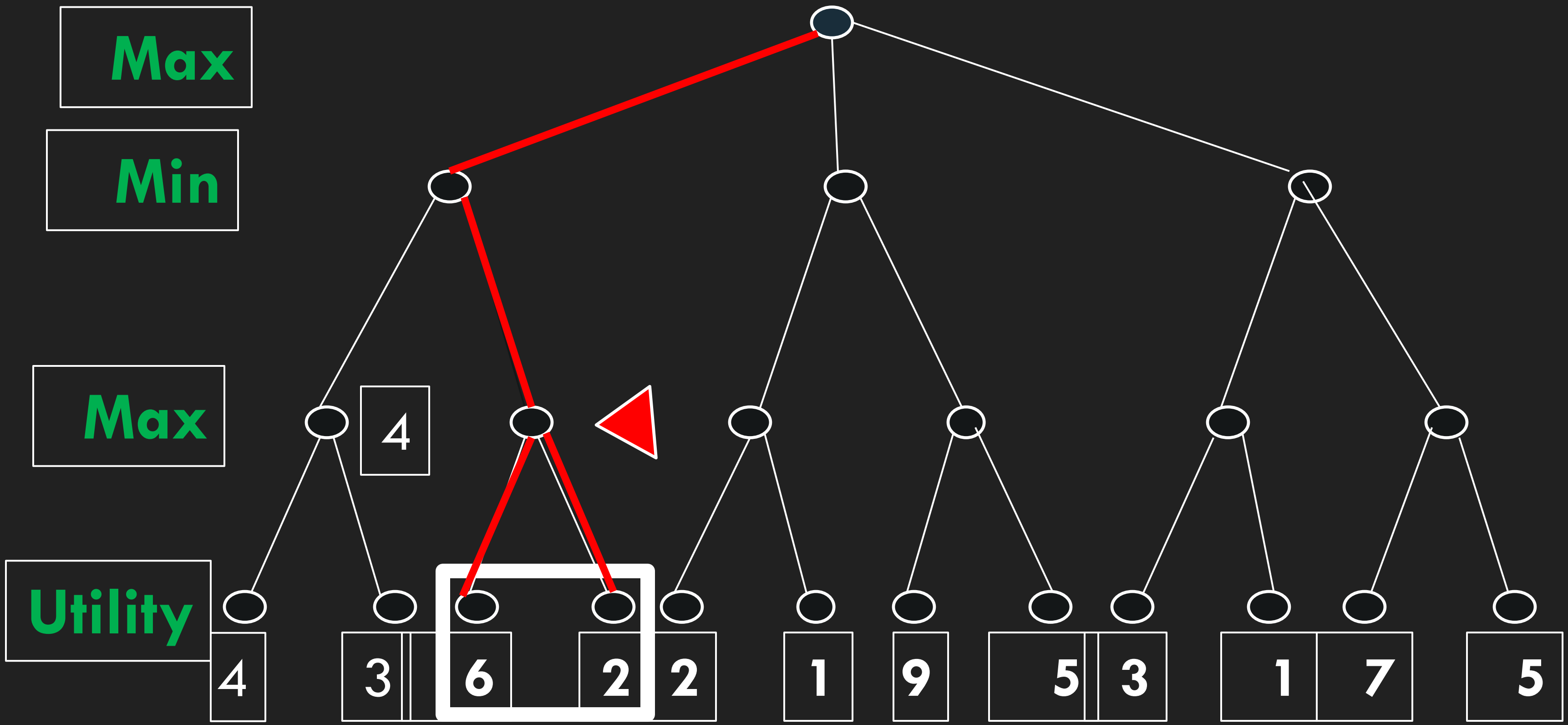
- Minimax is a decision rule algorithm, which is represented as a game-tree.
- It has applications in decision theory, game theory , statistics and philosophy.
- Minimax is applied in two player games. The one is the min and the other is the max player.
- By agreement the root of the game-tree represents the max player.
- It is assumed that each player aims to do the best move for himself and therefore the worst move for his opponent in order to win the game.

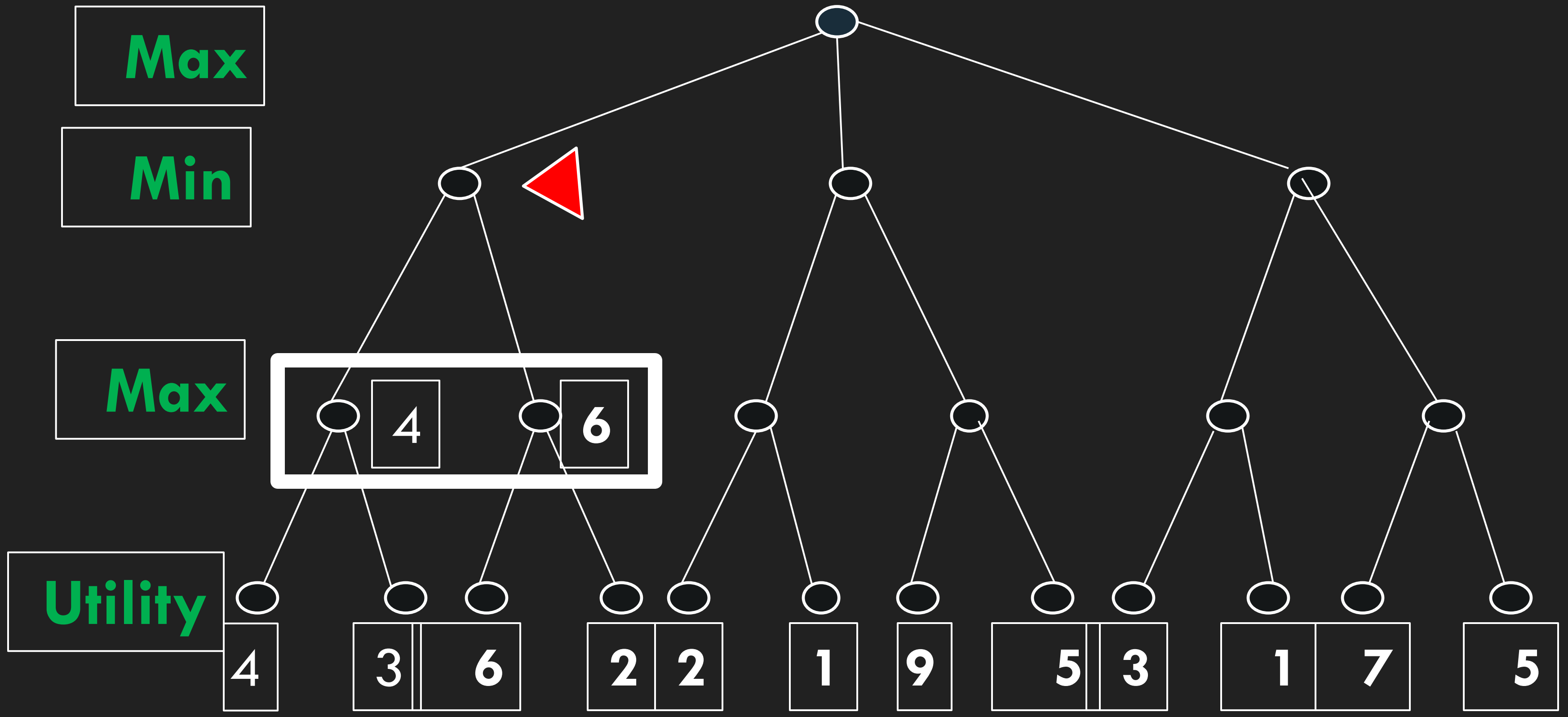
Example



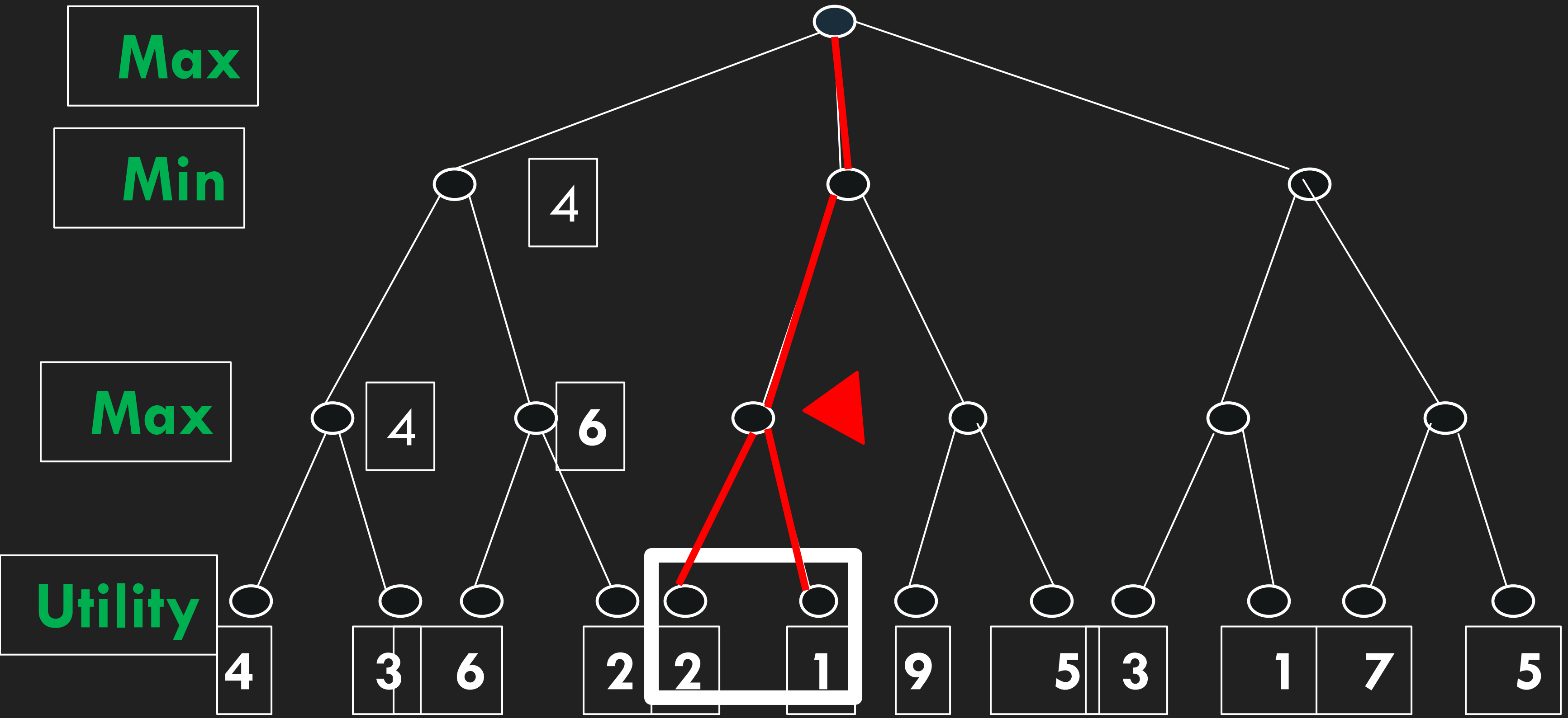


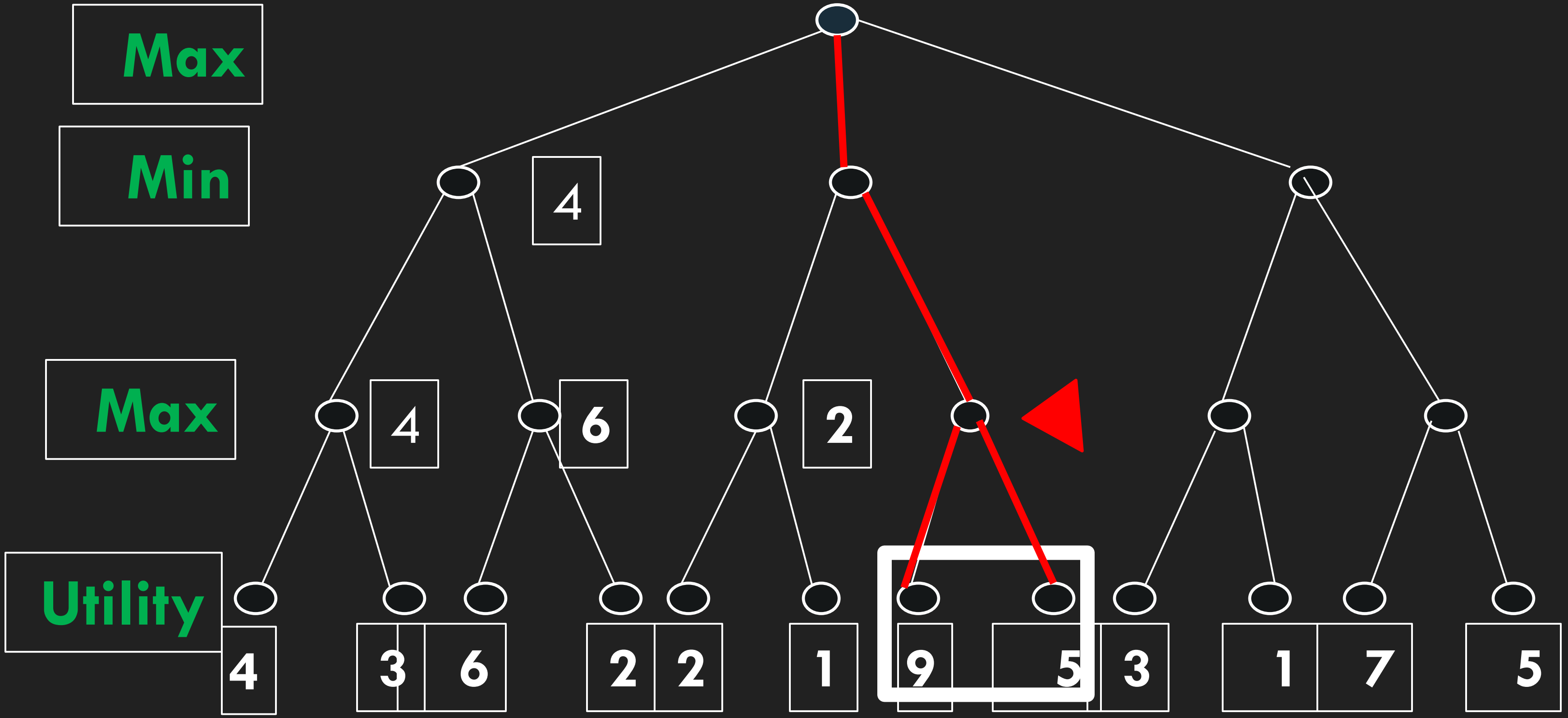
Max
Min
Max
Utility
4
3
6
2
2
1
9
5
3
1
7
5

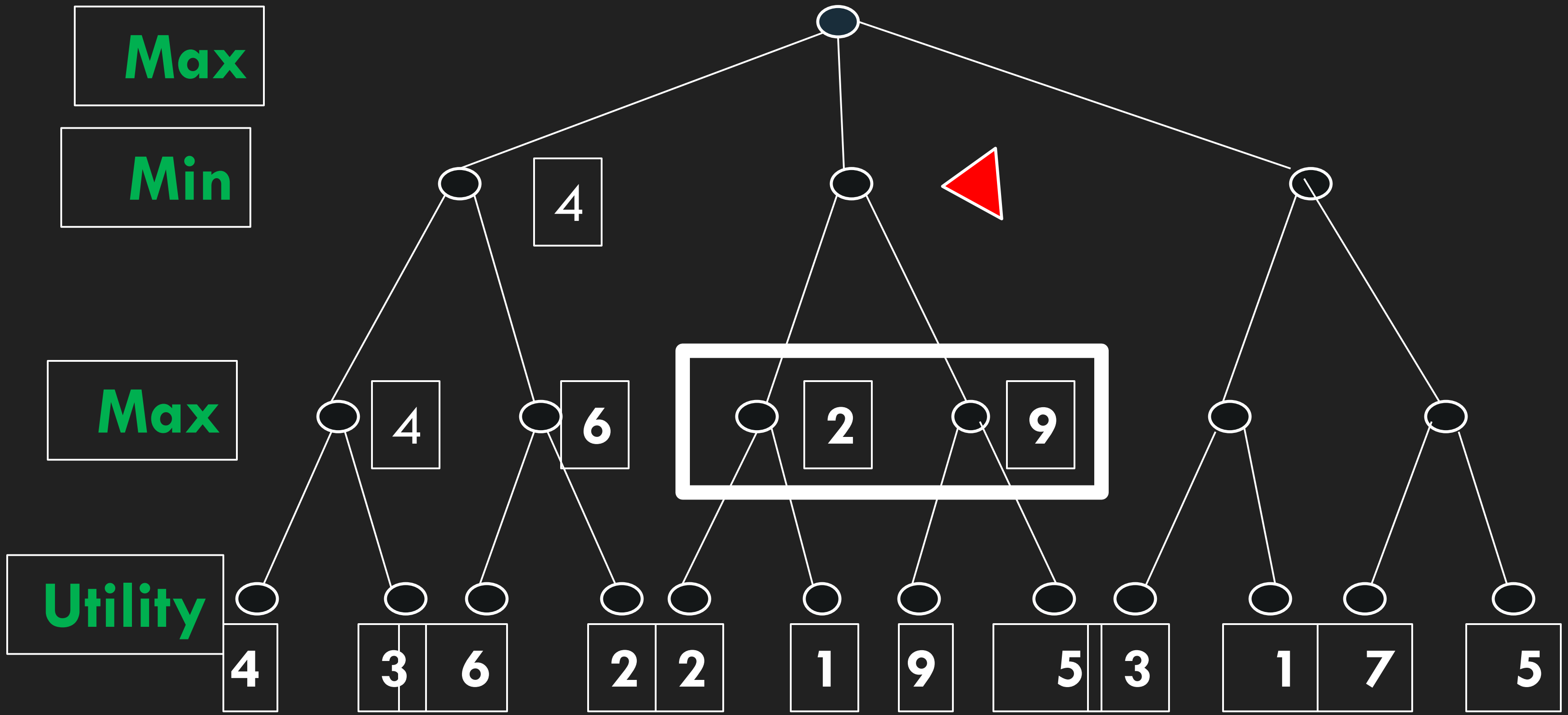

Max
Min
Max
4
Utility
4
3
6
2
2
1
9
5
3
1
7
5



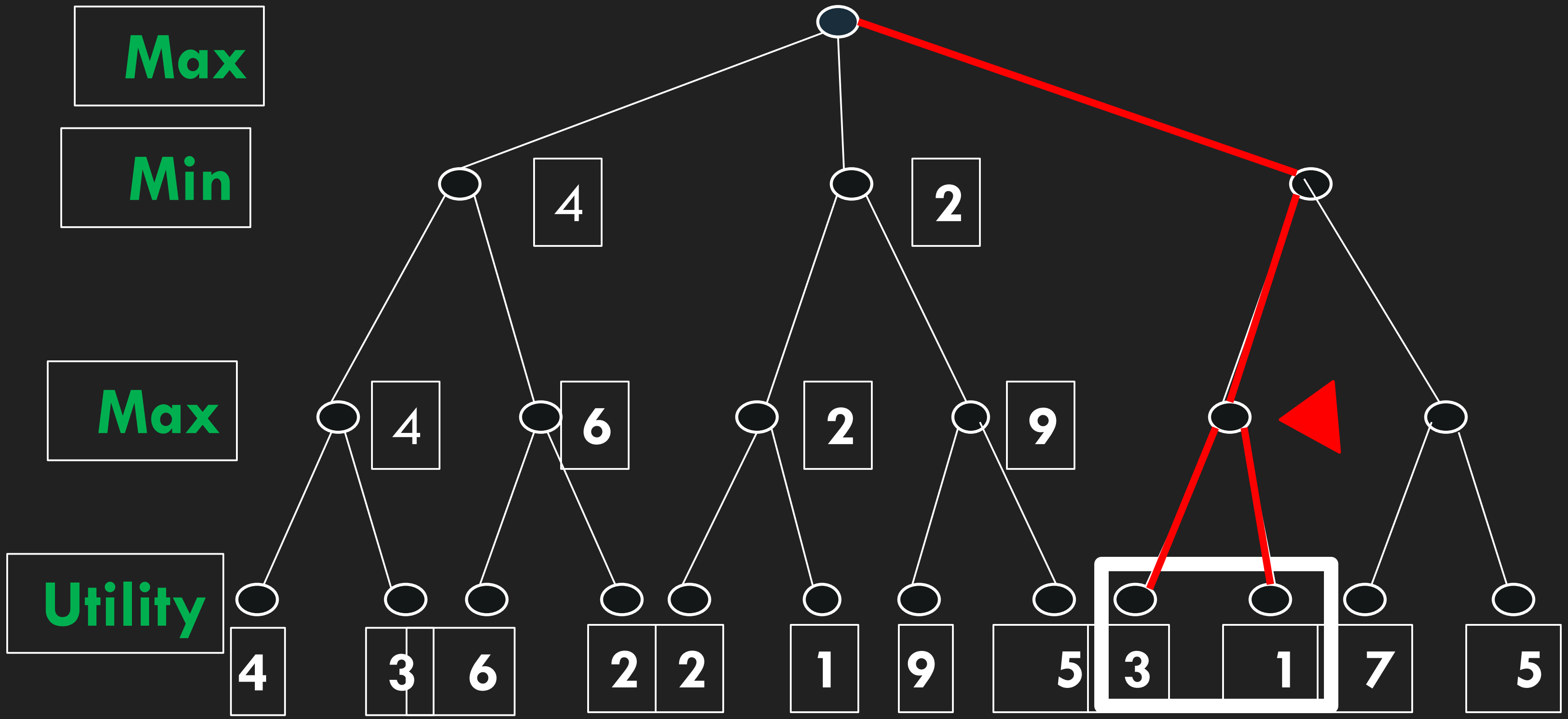
Max
Min
Max
Utility
4
6
4
3
6
2
2
1
9
5
3
1
7
5


Max
Min
Max
Utility
4
4
6
4
3
6
2
2
1
9
5
3
1
7
5

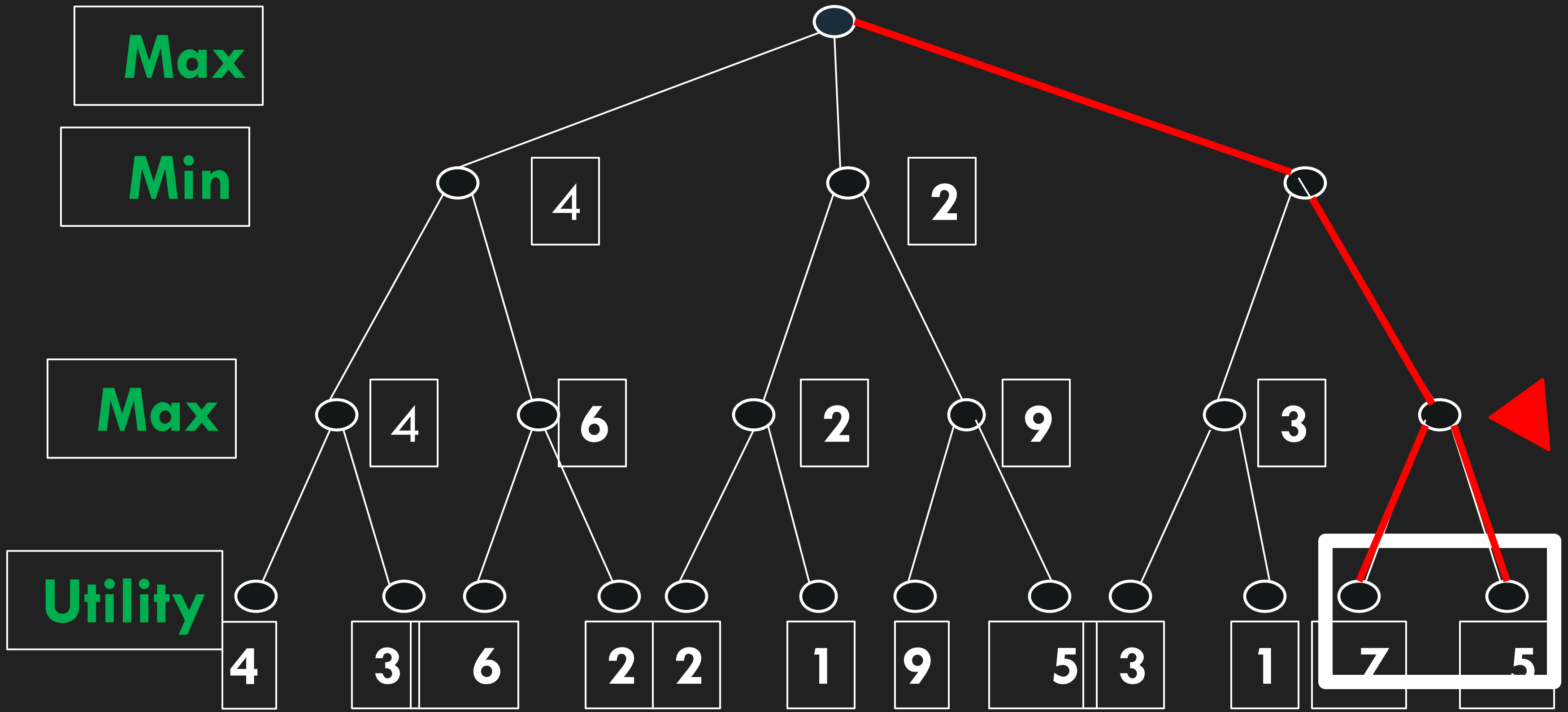

Max
Min
Max
Utility
4
4
6
2
4
3
6
2
2
1
9
5
3
1
7
5



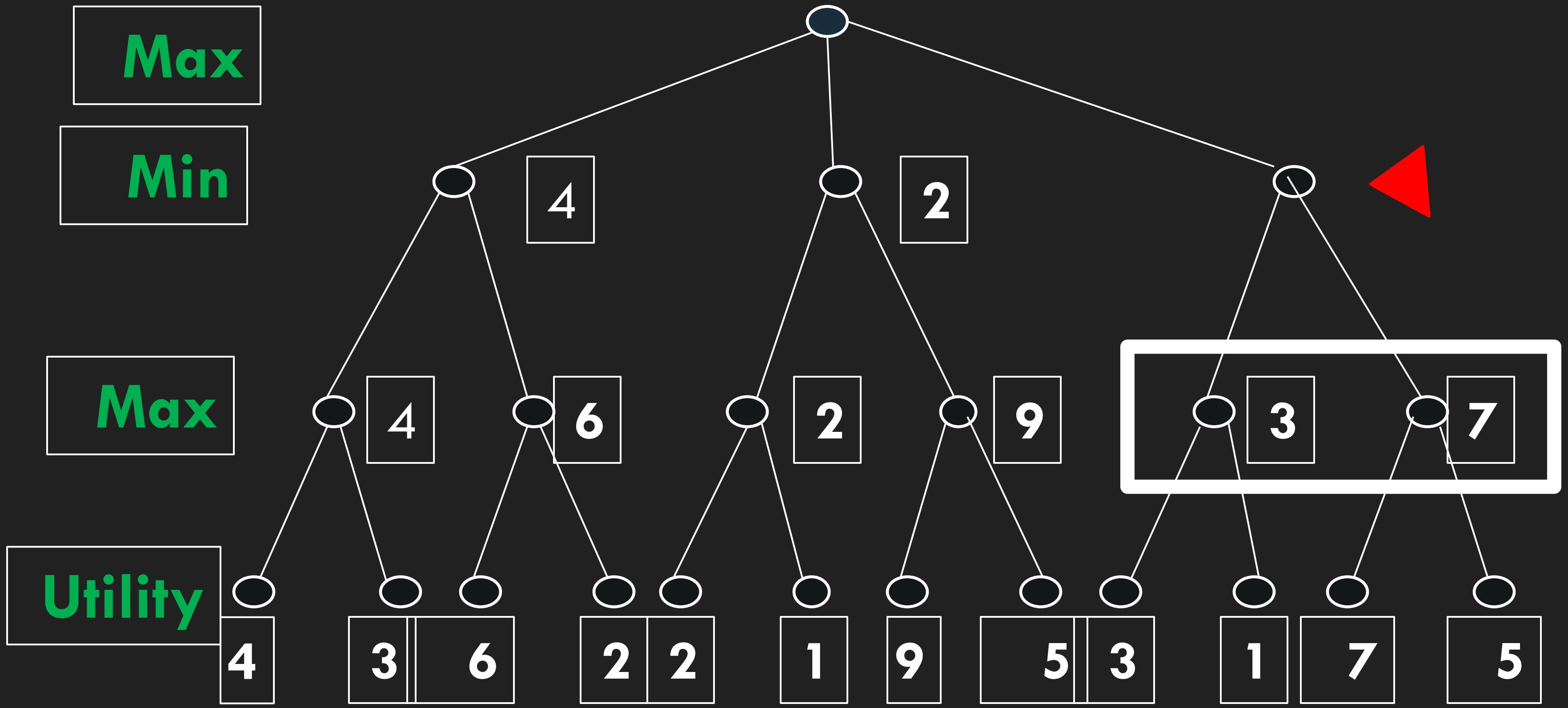
Max
Min
Max
Utility
4
4
6
2
9
4
3
6
2
2
1
9
5
3
1
7
5



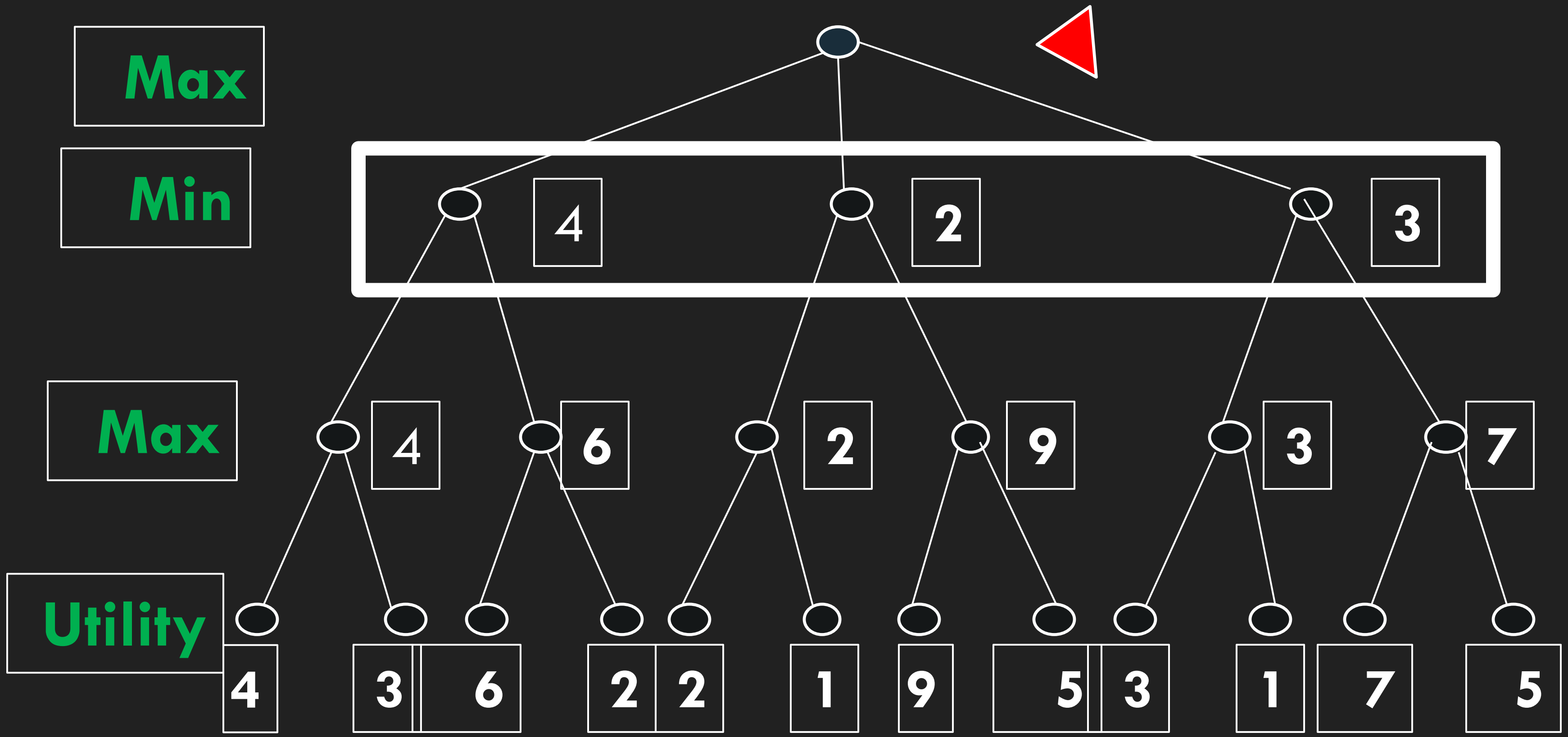
Max
Min
Max
Utility
4
2
4
6
2
9
4
3
6
2
2
1
9
5
3
1
7
5



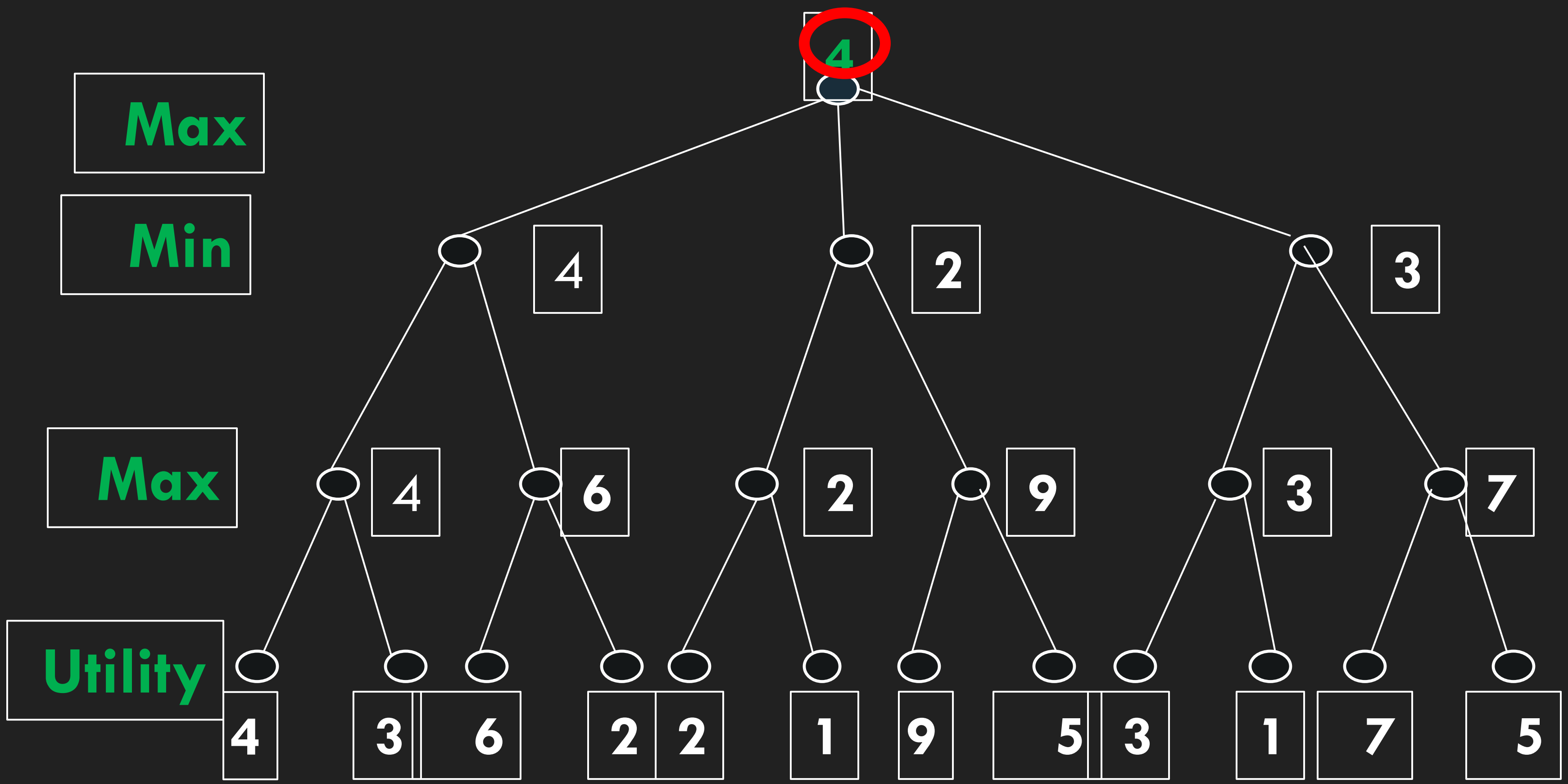
Max
Min
Max
Utility
4
2
4
6
2
9
3
4
3
6
2
2
1
9
5
3
1
7
5



Max
Min
Max
Utility
4
2
4
6
2
9
3
7
4
3
6
2
2
1
9
5
3
1
7
5

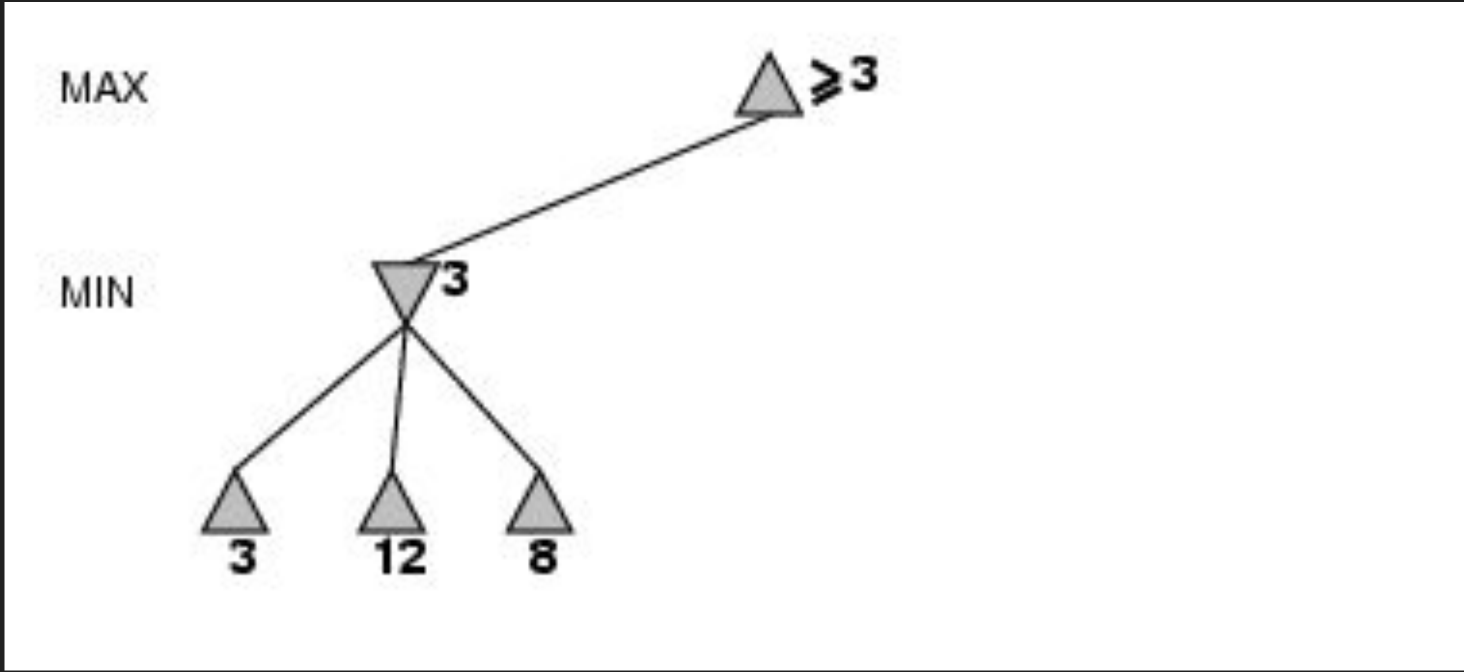


Max
Min
Max
Utility
4
2
3
4
6
2
9
3
7
4
3
6
2
2
1
9
5
3
1
7
5

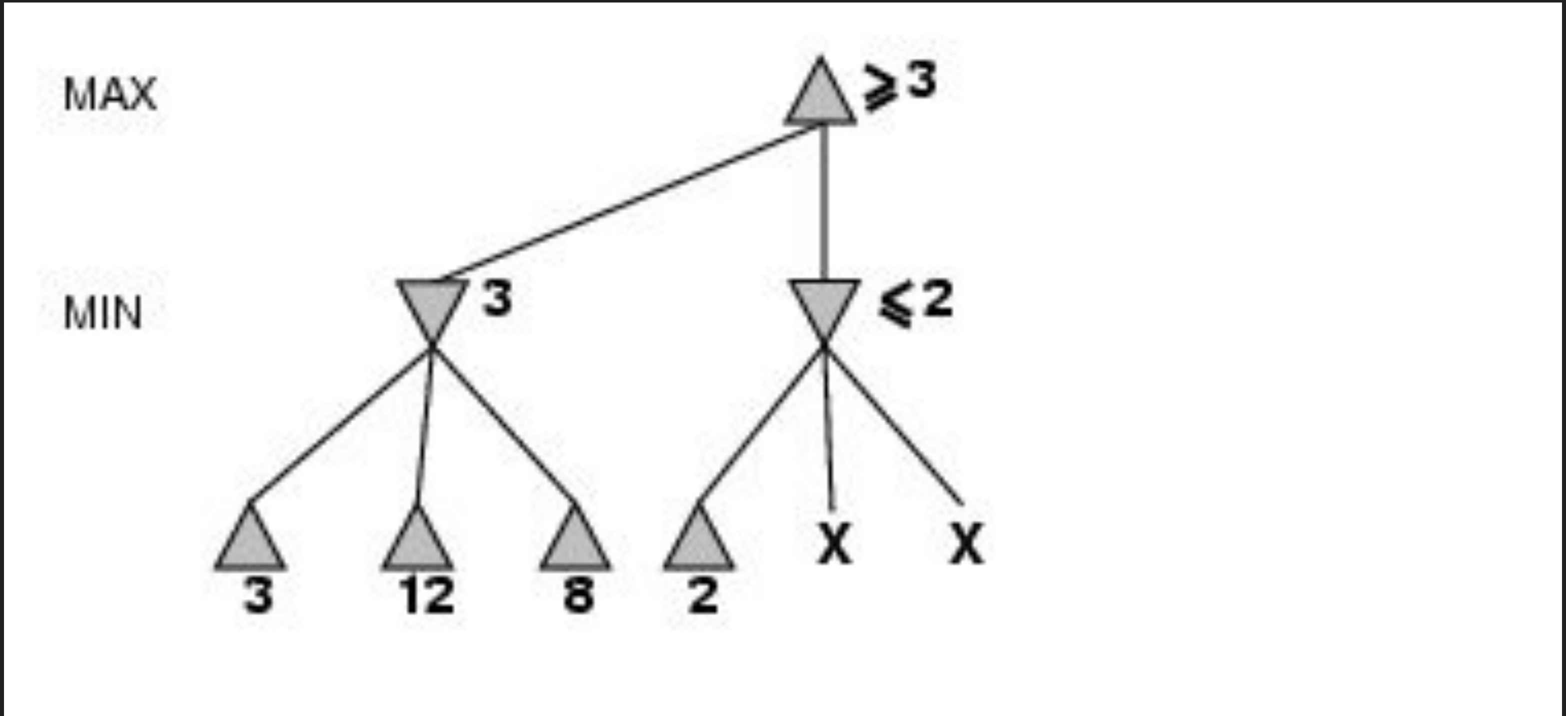


Max
Min
Max
Utility
4
4
2
3
4
6
2
9
3
7
4
3
6
2
2
1
9
5
3
1
7
5

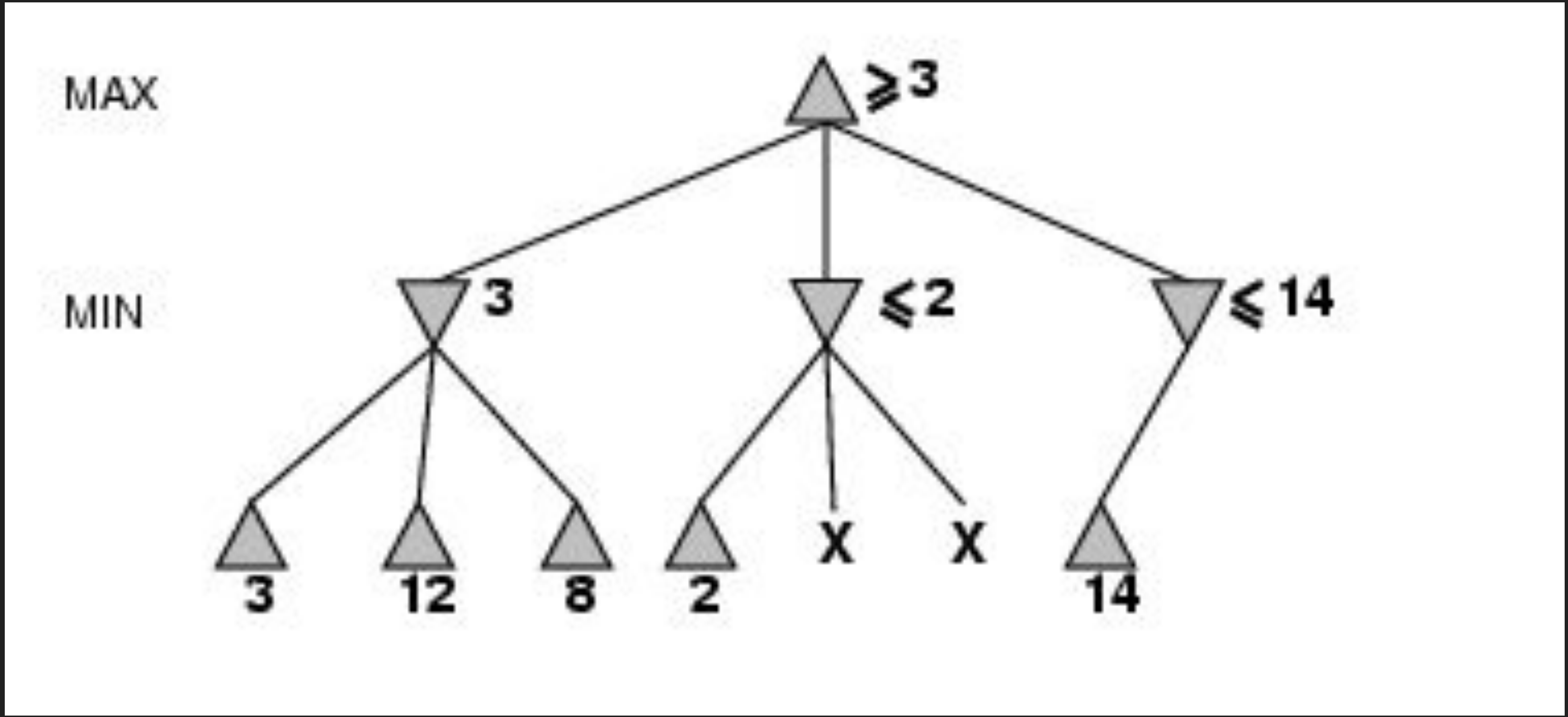
α-β pruning example



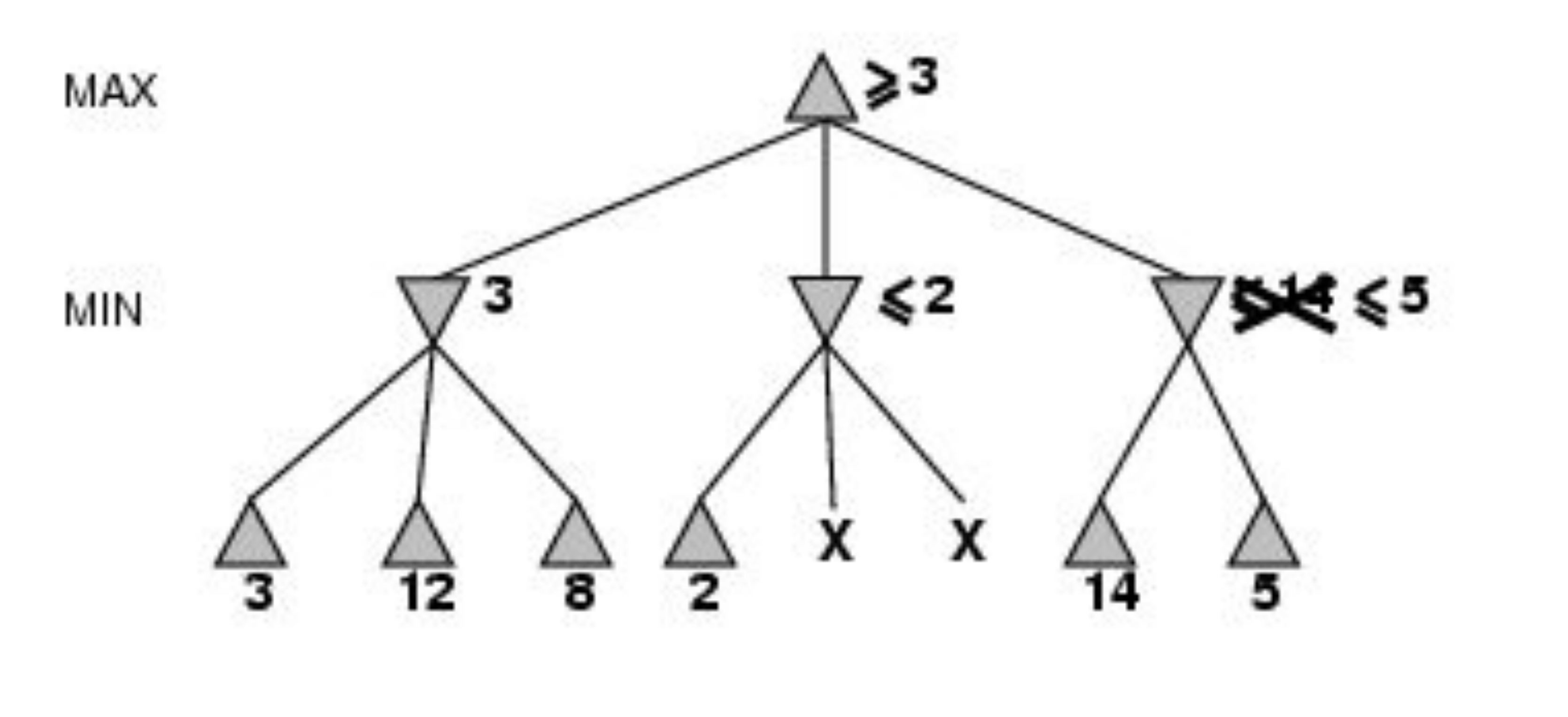
α-β pruning example



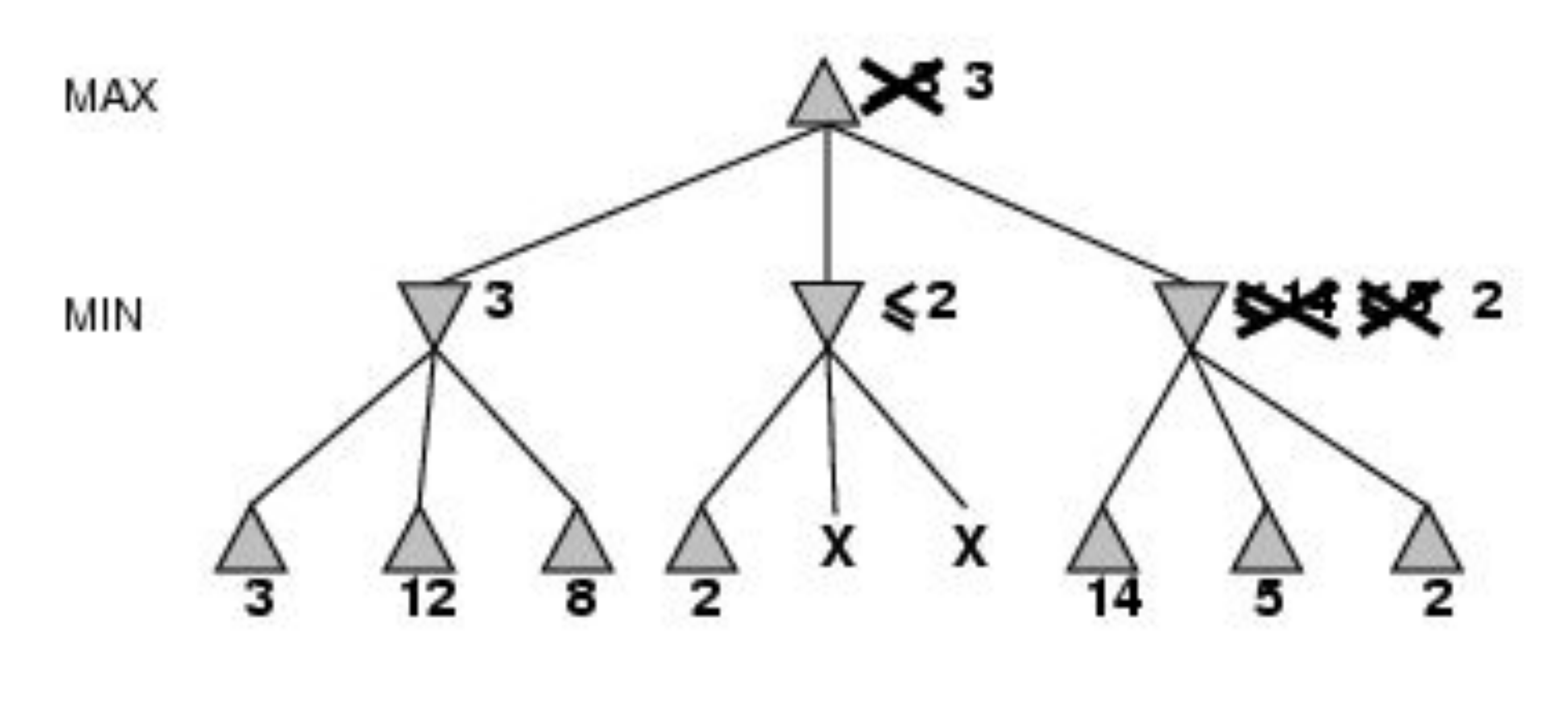
α-β pruning example



α-β pruning example



α-β pruning example

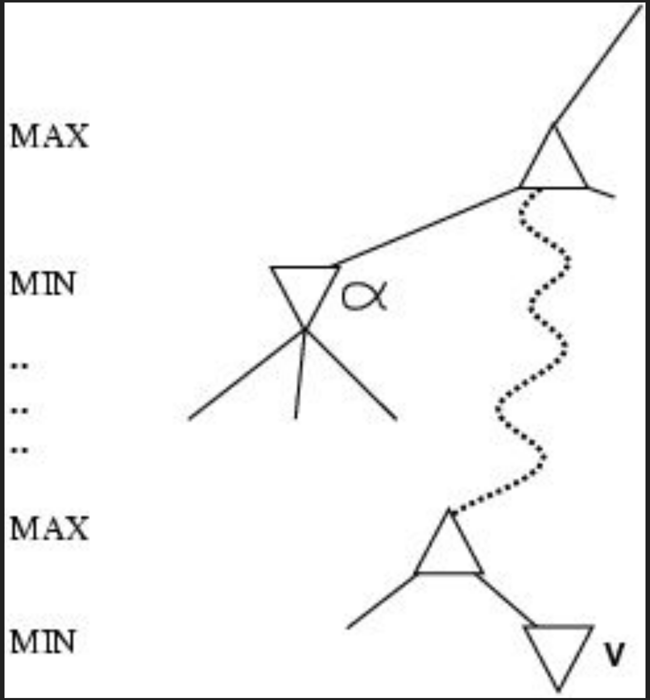


Properties of α-β

- 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
- 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

  $v \leftarrow$ MAX-VALUE(*state*, $-\infty$, $+\infty$)
  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*)
  $v \leftarrow -\infty$
  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

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*)
    $v \leftarrow +\infty$
    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

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)
- evaluation function
  = estimated desirability of position

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.

Evaluation functions

- First, the evaluation function should order the *terminal states in the same way as the* true utility function;
  Second, the computation must not take too long!
- Third, for nonterminal states, the evaluation function should be strongly correlated with the actual chances of winning.

Cutting off search

MinimaxCutoff is identical to *MinimaxValue* except

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

TERMINAL-TIEST-->**if CUTOFF-TEST(stated, depth) then return EVAL(state)**

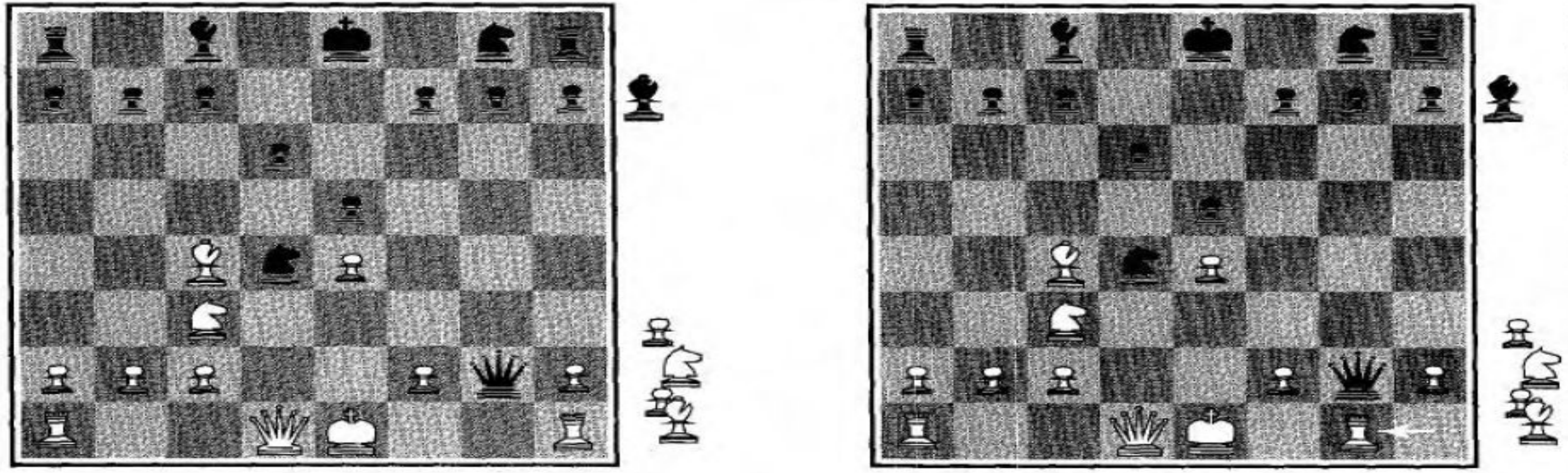
Does it work in practice?

$b^m = 10^6$, $b=35$ □ $m=4$

4-ply lookahead is a hopeless chess player!

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

Cutting off search

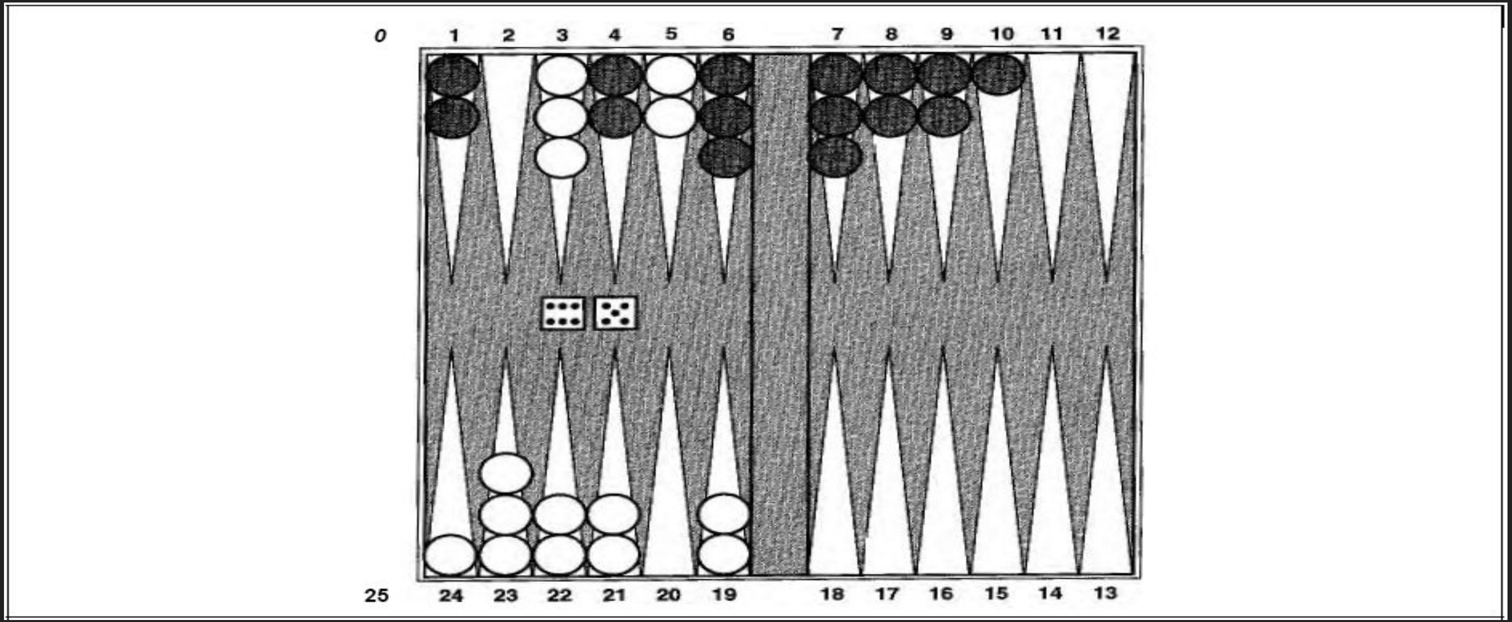


(a) White to move

(b) White to move

Figure 6.8 Two slightly different chess positions. In (a), black has an advantage of a knight and two pawns and will win the game. In (b), black will lose after white captures the queen.

Game Include an Element of Chance



Backgammon

Game Include an Element of Chance

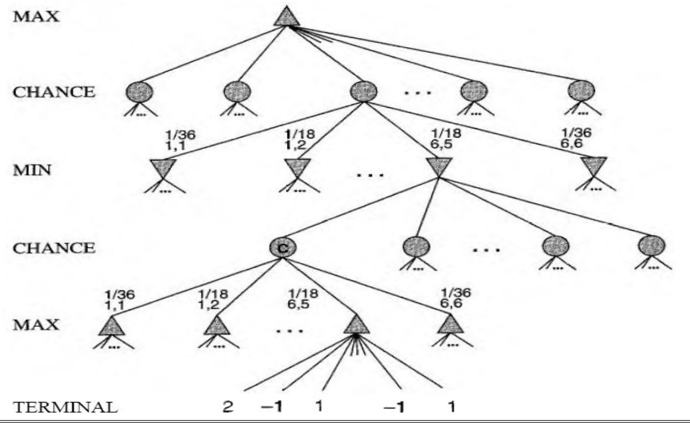


Figure 6.11 Schematic game tree for a backgammon position.

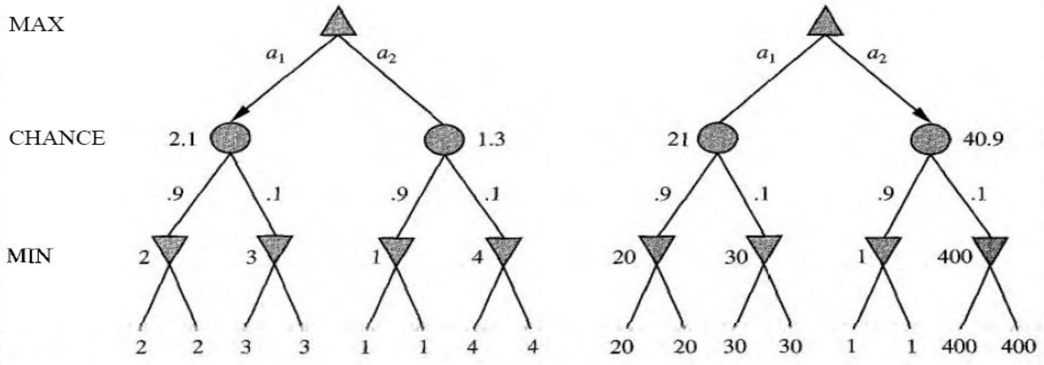


Figure 6.12 An order-preserving transformation on leaf values changes the best move.

$$\textsc{Expectiminimax}(n) = \begin{cases} \textsc{Utility}(n) & \text{if } n \text{ is a terminal state} \\ \max_{s \in Successors(n)} \textsc{Expectiminimax}(s) & \text{if } n \text{ is a } MAX \text{ node} \\ \min_{s \in Successors(n)} \textsc{Expectiminimax}(s) & \text{if n is a } MIN \text{ node} \\ \sum_{s \in Successors(n)} P(s) \,.\, \textsc{Expectiminimax}(s) & \text{if } n \text{ is a chance node} \end{cases}$$

Summary

- Games are fun to work on!
  They illustrate several important points about AI
  perfection is unattainable must approximate
- MiniMax and Alpha beta pruning