

# Games 0.1

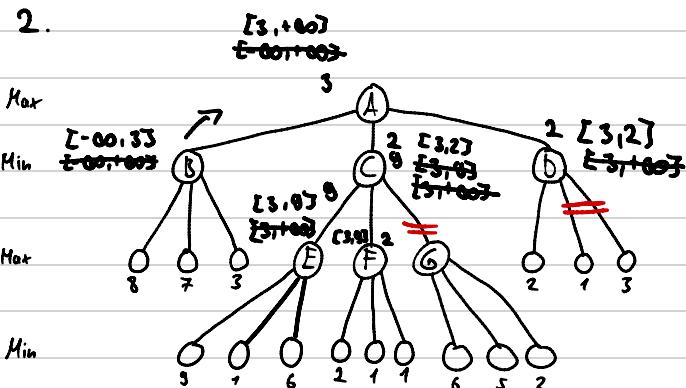
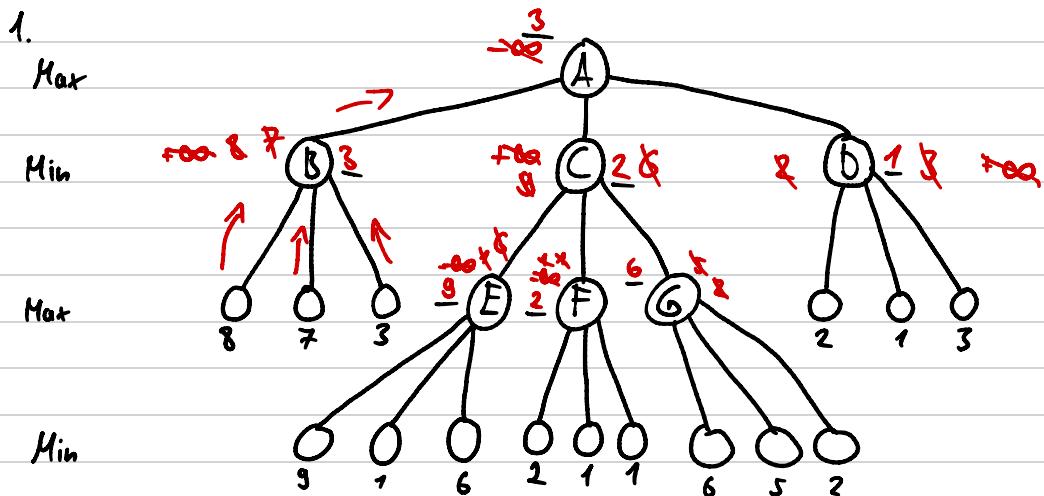
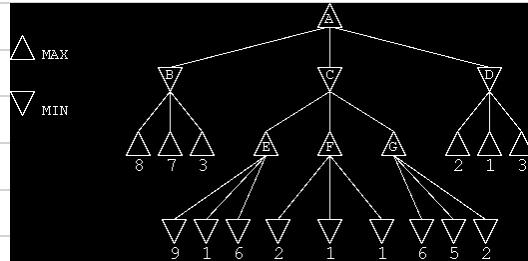
Geben Sie für den Spielbaum die Minimax Bewertungen an.  
 (Utility Werte sind schon gegeben bzw. Endzustände)

```
def Max-Value(state):
    if Terminal-Test(state): return Utility(state)

    v = -INF
    for (a, s) in Successors(state):
        v = MAX(v, Min-Value(s))
    return v

def Min-Value(state):
    if Terminal-Test(state): return Utility(state)

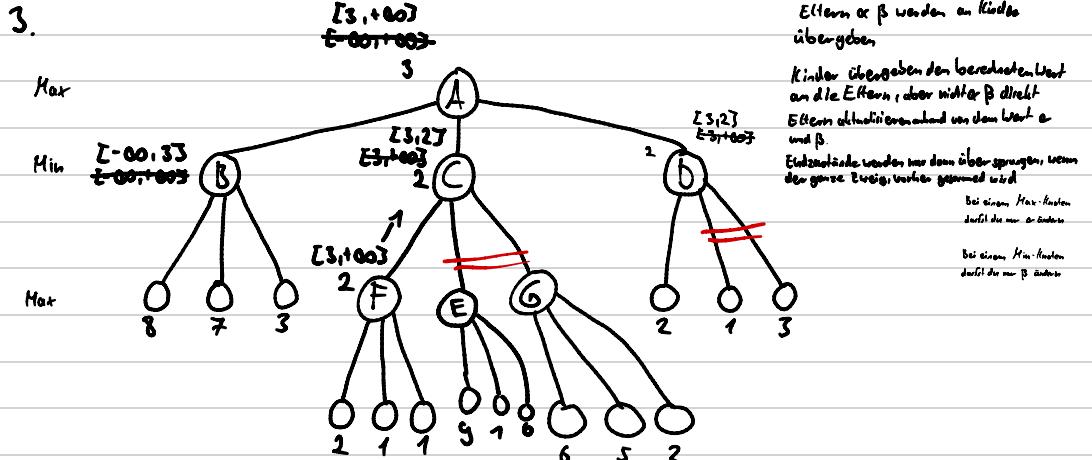
    v = +INF
    for (a, s) in Successors(state):
        v = MIN(v, Max-Value(s))
    return v
```



Mindest-Knotenknt  $\alpha \geq \beta$  (Min-Vergängen)  
 Aktueller Min-Knotenknt  $\beta \leq \alpha$  (Max-Vergängen)

$\overleftarrow{\text{Lose}} \rightarrow \overrightarrow{\text{Gewinnt}}$

$\overleftarrow{\text{Bisher besten Werte für}}  
Max$



### Nr 2.3

Gegeben: Leeres Tic-Tac-Toe Feld und Max/Min als Anfänger

Anzahl der verglichenen Knoten: ca. 550.000 (ohne Pruning) und ca. 18.000 (mit Pruning)

# # #  
# # #  
# # #

Gegeben: 6 freie Felder und Min im Zug

Anzahl der verglichenen Knoten: 1097 (ohne Pruning) und 336 (mit Pruning)

X O X  
# # #  
# # #

Gegeben: 8 freie Felder, starke Positionierung von Max und Min Zug

Anzahl der verglichenen Knoten: 55505 (ohne Pruning) und 2316 (mit Pruning)

# # #  
# X #  
# # #

- Erhebliche Reduzierung der verglichenen

Gegeben: 8 freie Felder und Min Zug

Anzahl der verglichenen Knoten: 55705 (ohne Pruning) und 2338 (mit Pruning)

X # #  
# # #  
# # #

- Erhebliche Reduzierung der verglichenen

(kürzester Spielbaum)

Je weniger freie Felder, desto weniger miteinander zu vergleichende Knoten. Mit Pruning erhebliche Reduzierung der Anzahl miteinander zu vergleichender Knoten. Kleine Abweichungen basierend auf Positionierung, bei gleicher Anzahl freier Felder.

## Games . 03 (Ziel: Ergebnis, wenn beide Spieler perfekt spielen)

Pseudocode:

```
function Minimax (state) {
    if Terminal-Test(state): return Utility (state)
```

$$\text{bestV} = -\text{INF}$$

for ( $a_{1,2}$ ) in Successors (state):

$$\text{bestV} = \max (\text{bestV}, -(\text{Minimax}(s)))$$

return bestV

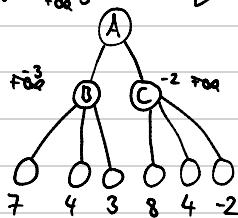
$a = 1$  } Parameterübergabe die bestimmt, ob Max oder Min abwechselnd im Zug ist  
 $a = -1$  }

Minimax (vereinfacht)

Rückwärts wird es wieder begiert bzw.  $-(-3)$  und  $-(-2)$

Max

Max  
(simuliert aber Min durch Verzweigungen)



$\cdot (-)$

$$(-1) \cdot 7 = -7$$

$$(-1) \cdot 4 = -4$$

$$(-1) \cdot 2 = -2 \leftarrow \text{Max bzw. Minwert}$$

$$(-1) \cdot 8 = -8$$

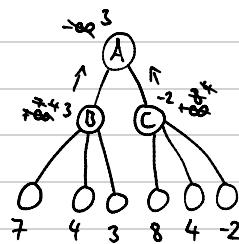
$$(-1) \cdot 4 = -4$$

$$(-1) \cdot (-2) = 2$$

Minimax (normal)

Max

Min



## Games 0.4

Definieren:

$X_n$ : Die Anzahl der Reihen, Spalten oder Diagonalen mit exakt  $n$  X's und keinem O

$O_n$ : Die Anzahl der Reihen, Spalten oder Diagonalen mit exakt  $n$  O's und keinem X

Utility function: Wert +1 jeder Position mit  $X_3=1$  und -1 zu jeder Position  $O_3=1$

Alle anderen Endzustände (Terminalzustände) haben die utility 0 (null)

Für nicht-Terminalpositionen benutzen wir die Funktion:

Zielzustände bewerten ohne bis zum Ende gehen zu müssen.

$$\text{Eval}(s) = 3X_2(s) + X_1(s) - (3O_2(s) + O_1(s))$$

3 Endzustände:

$$\begin{array}{ccc} X & O & X \\ XX & O \\ X & \# & O \end{array} \quad \begin{array}{ll} X_1=0 & O_2=0 \\ X_2=0 & O_1=0 \\ \text{Eval}(s) = 3 \cdot 0 + 0 - (3 \cdot 0 + 0) = 0 \end{array}$$

$$\begin{array}{ccc} X & X & O \\ 0 & 0 & 0 \\ X & \# & X \end{array} \quad \begin{array}{ll} X_1=0 & O_2=0 \\ X_2=1 & O_1=0 \\ \text{Eval}(s) = 3 \cdot 1 + 0 - (3 \cdot 0 + 0) = 3 \end{array}$$

$$\begin{array}{ccc} X & \# & X \\ \# & 0 & X \\ 0 & 0 & X \end{array} \quad \begin{array}{ll} X_1=0 & O_2=1 \\ X_2=1 & O_1=0 \\ \text{Eval}(s) = 3 \cdot 1 + 0 - 3 \cdot 1 + 0 = 0 \end{array}$$

3 Zwischenzustände:  $\text{Eval}(s) = 3X_2(s) + X_1(s) - (3O_2(s) + O_1(s))$

$$\begin{array}{ll} \# \# \# & X_1 = 3 \quad O_2 = 0 \\ \# X \# & X_2 = 1 \quad O_1 = 1 \\ 0 \# X & \text{Eval}(s) = 3 \cdot 1 + 3 - (3 \cdot 0 + 1) = 5 \end{array}$$

$$\begin{array}{ll} X \# 0 & X_1 = 1 \quad O_2 = 1 \\ \# 0 \# & X_2 = 0 \quad O_1 = 3 \\ \# \# \# & \text{Eval}(s) = 3 \cdot 0 + 1 - (3 \cdot 1 + 3) = -5 \end{array}$$

$$\begin{array}{ll} X \# \# & X_1 = 3 \quad O_2 = 0 \\ \# \# \# & X_2 = 0 \quad O_1 = 0 \\ \# \# \# & \text{Eval}(s) = 3 \cdot 0 + 3 - (3 \cdot 0 + 0) = 3 \end{array}$$

$$\begin{array}{ll} \# \# \# & X_1 = 4 \quad O_2 = 0 \\ \# X \# & X_2 = 0 \quad O_1 = 0 \\ \# \# \# & \text{Eval}(s) = 3 \cdot 0 + 4 - (3 \cdot 0 + 0) = 4 \end{array}$$

Mit der Einwertungsfunktion können Zwischenzustände gut ausgewertet werden. Sie berücksichtigt und bevorzugt Positionierungen wie die Mitte. Zudem zeigt sie auch wer dominiert im Tic-Tac-Toe. Die Endzustände bewertet sie jedoch nicht angemessen. Sie bewertet den ersten Endzustand im Beispiel mit 0, was ein Unentschieden entsprechen würde aber keins ist. Dieser Fehler wird auch wiederholt.

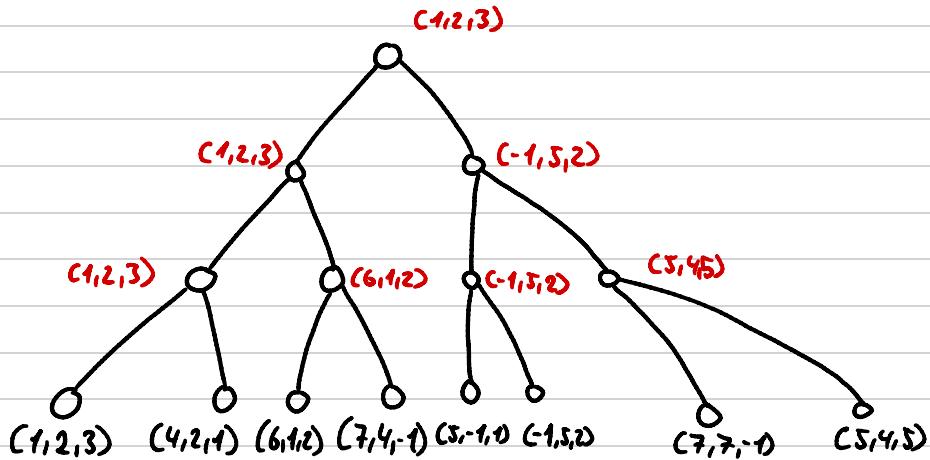
## Games 0.5

Spieler 1

Spieler 2

Spieler 3

Spieler 1



Wollen immer maximieren!

Position des Spielers bestimmt welches Element i im Tripel berücksichtigt wird.