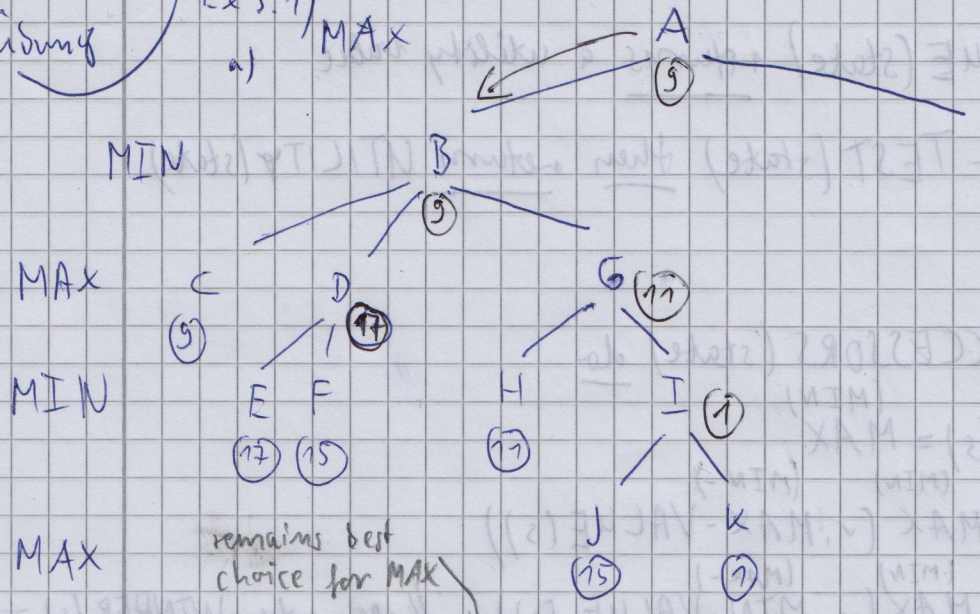
üB3

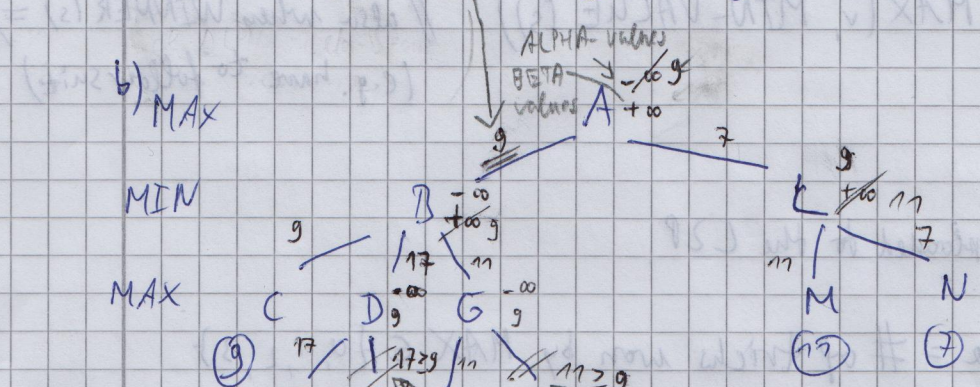
AI Lösung

Ex 3.1)

a)



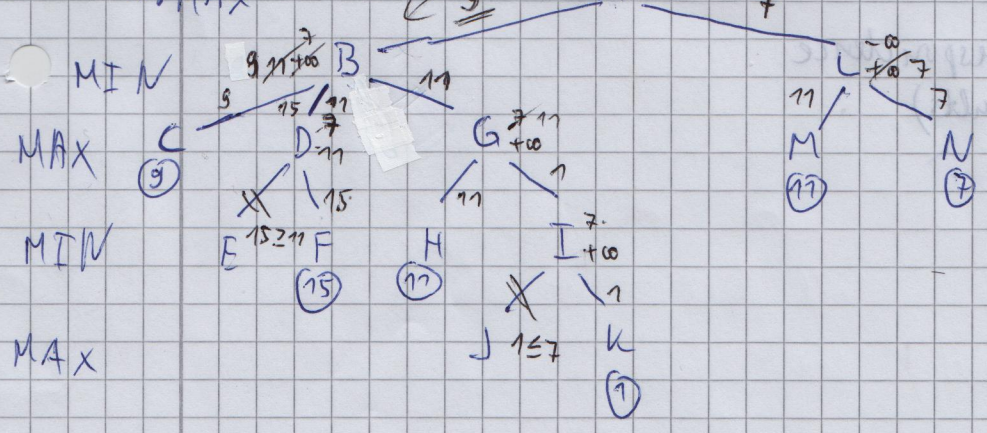
b)



(cf. ALPHA-BETA-Search version 2)

pruned: F, I, J, K

c)



pruned: E, J

Complexity for Minmax: $O(b^d)$

α-β with optimal ordering: $O(b^{d/2})$

α-β on average: $O(b^{3d/4})$

Ex 3.2) a) (MIN-)

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

if TERMINAL-TEST(state) then return UTILITY(state)

v ← (+∞) −∞

for a,s in SUCCESSORS(state) do

if WINNER(s) = MAX (MIN)

then v ← MAX(v, MAX-VALUE(s)) (MIN) (MIN-)

else v ← MAX(v, MIN-VALUE(s)) (MIN) (MAX-1) // also when WINNER(s) = ∅ (e.g. have to follow suit)

return v

b) game-tree will be uploaded to the L2P

Note: utility value = # of tricks won by MAX ∈ {0, 1, 2, 3}

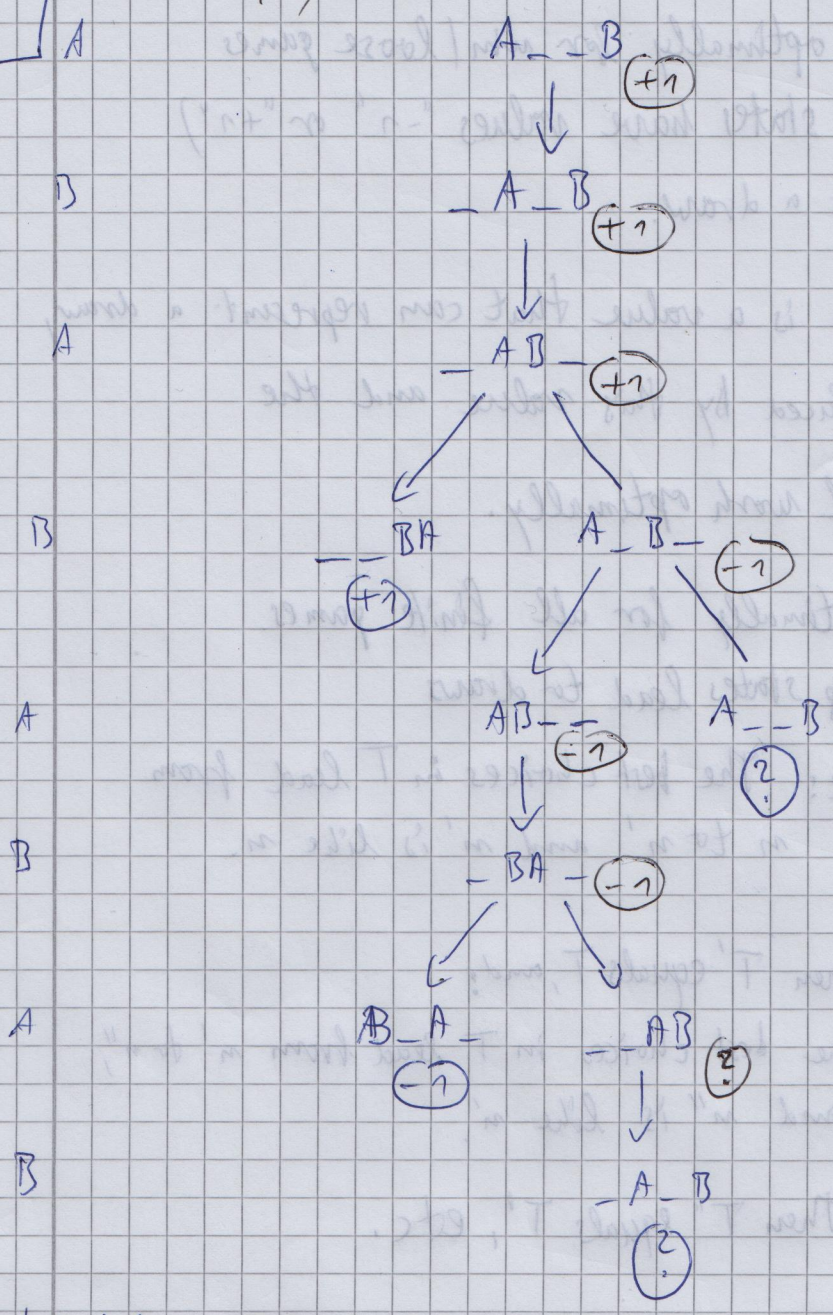
Alternative: utility value:

(# tricks won by MAX) − (# tricks won by MIN) ∈ {−3, −1, 1, 3}

⇒ one-to-one correspondence
(yield same results)

AI — Übung — (Ex 3.3) a) / b)

Übung 3



b) pass the (?) from bottom upwards

c) trivial answers:

- standard minimax cannot handle questionmarks
- if loop states would not have been considered as terminal states, then minimax would run into an infinite path (depth-first)

fix: keep track of all (loop) states, and define max and min for each non-empty subset of $\{-1; ?; +1\}$:

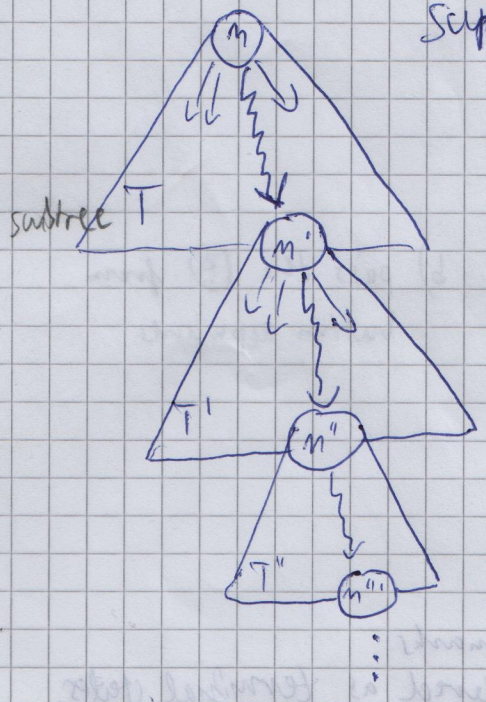
	$\{-1, ?, +1\}$	$\{+1, ?\}$	$\{+1, -1\}$	$\{?, -1\}$	$\{+1\}$	$\{?\}$	$\{-1\}$
A = MAX	+1	+1	+1	?	+1	?	-1
B = MIN	-1	?	-1	-1	+1	?	-1

Note: Exactly the same result will be obtained if "?" is replaced by "± 0".

This will always work optimally for win/loose games (i.e. when terminal states have values "-1" or "+1") because "±0" represents a draw.

More general: If there is a value that can represent a draw, then "?" can be replaced by this value and the standard minimax will work optimally.

"Proof": Minimax works optimally for all finite games.
⇒ only to show: loop states lead to draws



Suppose: The best choices in T lead from n to n' and n' is like n.

Then T' equals T, and:
the best choices in T lead from n' to n'', and n'' is like n'.

Then T'' equals T', etc.

⇒ If n' is reached, then it will lead to an infinite path.
Infinite paths are draws (nobody wins).
⇒ Loop states lead to draws.

"□"