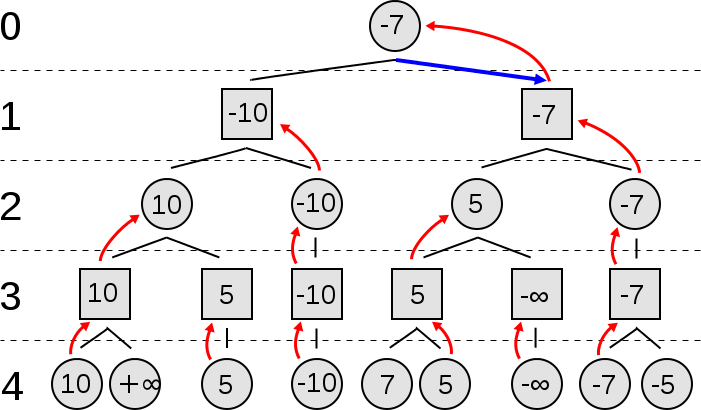
Minimax – Algorithmus

* Berechnet höchstmöglichen Gewinn
* Unabhängig von Spielweise des Gegners
* Bewertung von Spielzügen : +1 A gewinnt, -1 B gewinnt, 0 bei unentschieden
* Je tiefer der Suchbaum desto perfekter das Spiel

Das Bild rechts zeigt einen einfachen Baum mit Suchtiefe 4. Spieler A ist am Zug.

Die Knoten der Ebenen 0 und 2 entsprechen Spielsituationen, in denen Spieler A am Zug ist. Hier wird jeweils die Bewertungsfunktion der untergeordneten Knoten maximiert, d. h. der für Spieler A günstige Zug ausgewählt und dessen Wert dem Elternknoten zugewiesen.

Die Knoten der Ebenen 1 und 3 entsprechen Spielsituationen, in denen Spieler B am Zug ist. Hier wird jeweils die Bewertungsfunktion der untergeordneten Knoten minimiert, d. h. der für Spieler B günstigste Zug ausgewählt und dessen Wert dem Elternknoten zugewiesen.

Der Algorithmus beginnt unten bei den Blättern und geht dann nach oben bis zur Wurzel. In Ebene 3 wählt der Algorithmus den kleinsten Wert der Kindknoten und weist diesen dem Elternknoten zu (es wird minimiert). In Ebene 2 wird dann der jeweils größte Kindknoten dem Elternknoten zugewiesen (es wird maximiert). Dies wird abwechselnd so lange durchgeführt, bis die Wurzel erreicht ist. Der Wurzel wird der Wert des größten Kindknotens zugewiesen. Dabei handelt es sich dann um den Zug, der gespielt werden soll.

Pseudo-Code

<https://de.wikipedia.org/wiki/Minimax-Algorithmus>

Optimierte Variante – Negamax

<https://en.wikipedia.org/wiki/Negamax>

Einfache Prolog-Implementierung

<http://www.emse.fr/~picard/cours/ai/minimax/>

Bewertung Spielstand

<http://tim.hibal.org/blog/playing-checkers-with-minimax-continued/>

<http://stackoverflow.com/questions/2743752/static-evaluation-function-for-checkers>