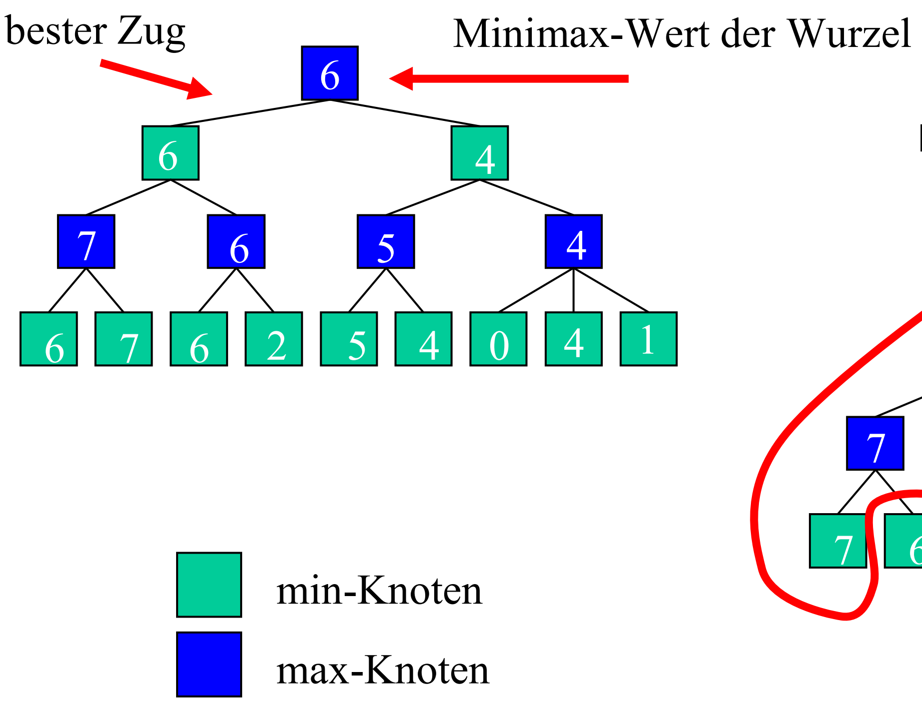
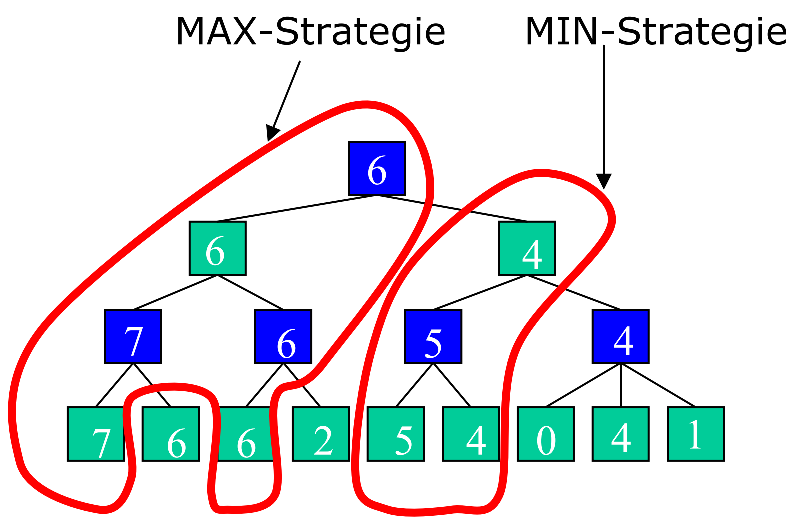
TECHNISCHE
UNIVERSITÄT
DARMSTADT

Computerschach
Grundlagen I

Was macht ein Schachprogramm eigentlich?

(II) Vorausschau





Minimax-Prinzip

Definition: Im folgenden ist G = (T,h) ein **Spielbaum**, wenn T = (V,E) ein Baum ist und h : L(G) → {0,1} eine Funktion ist, die Blattknoten des Spielbaums G auf Zahlen abbildet. L(G) bezeichnet hier die Menge der Blattknoten von G.

Definition: Es gebe 2 Spieler MIN und MAX. MAX besitzt das Zugrecht auf den geraden Ebenen des Spielbaums, MIN auf den anderen. Hierdurch wird eine Spielerfunktion definiert: **p : V → {MAX, MIN}**

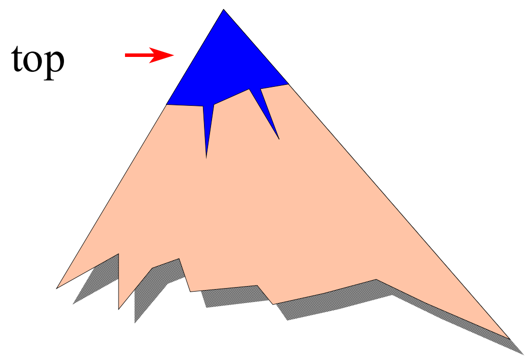
Definition: Sei G = ((V,E),h) ein Spielbaum. Sei v ∈ V ein Knoten des Spielbaums G. Die Funktion **minimax: V → {0,1}** ist induktiv wie folgt definiert:

$$\text{minimax}(v) := \begin{cases} h(v), & \text{falls } v \in L(G) \\ \max\{\text{minimax}(v' \mid (v,v') \in E\}, & \text{falls } p(v) = \text{MAX} \\ \min\{\text{minimax}(v') \mid (v,v') \in E\}, & \text{falls } p(v) = \text{MIN} \end{cases}$$

Der Minimaxwert des Knotens v ist minimax(v). Der Minimaxwert der Wurzel eines Spielbaums G wird mit minimax(G) bezeichnet.

Schnell, schnell, schnell!

Der kritische Punkt ist die intelligente Vorausschau.



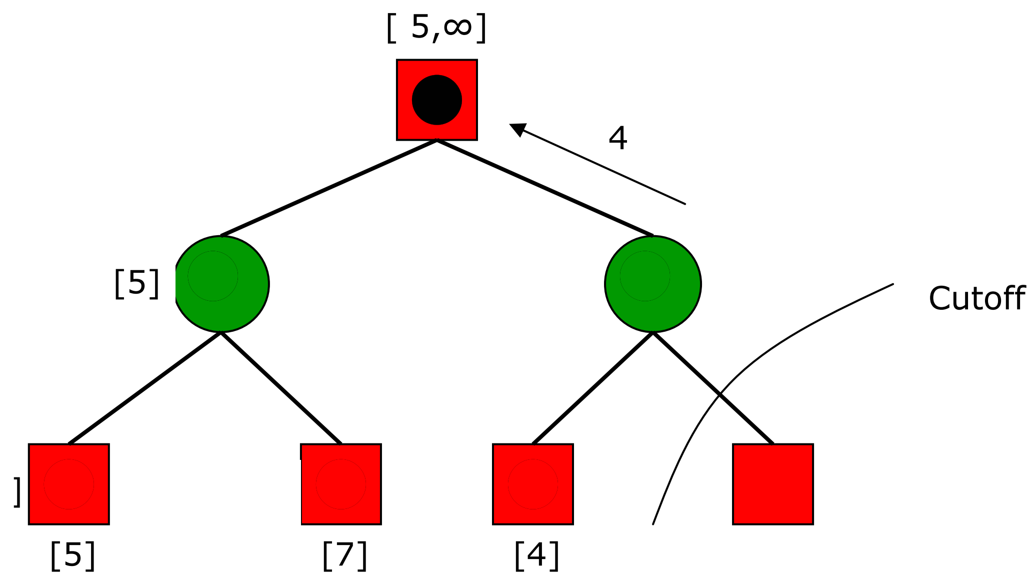
- **Man wähle einen geeigneten Teilbaum, der die aktuelle Stellung enthält. So groß wie möglich.**
- **Weise den künstlichen Endstellungen heuristische Werte zu!**
- **Werte aus!**

50 Jahre währende Erfahrung zeigt:

Der Spielbaum verhält sich wie ein Fehlerfilter!

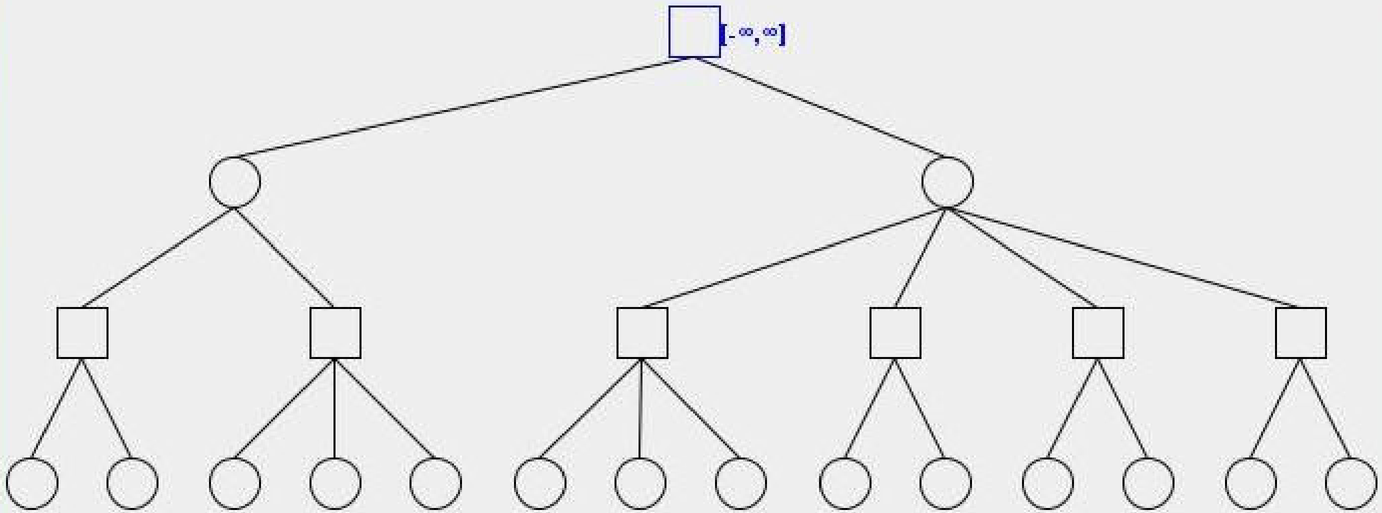
*Je größer der Baum, desto besser der Filter. Der **Alphabeta-Algorithmus** spielt dabei eine wichtige Rolle.*

Alphabeta-Algorithmus



Miniaturbeispiel

Alphabeta-Algorithmus



Weiter

Spielbaum 1

Code animieren

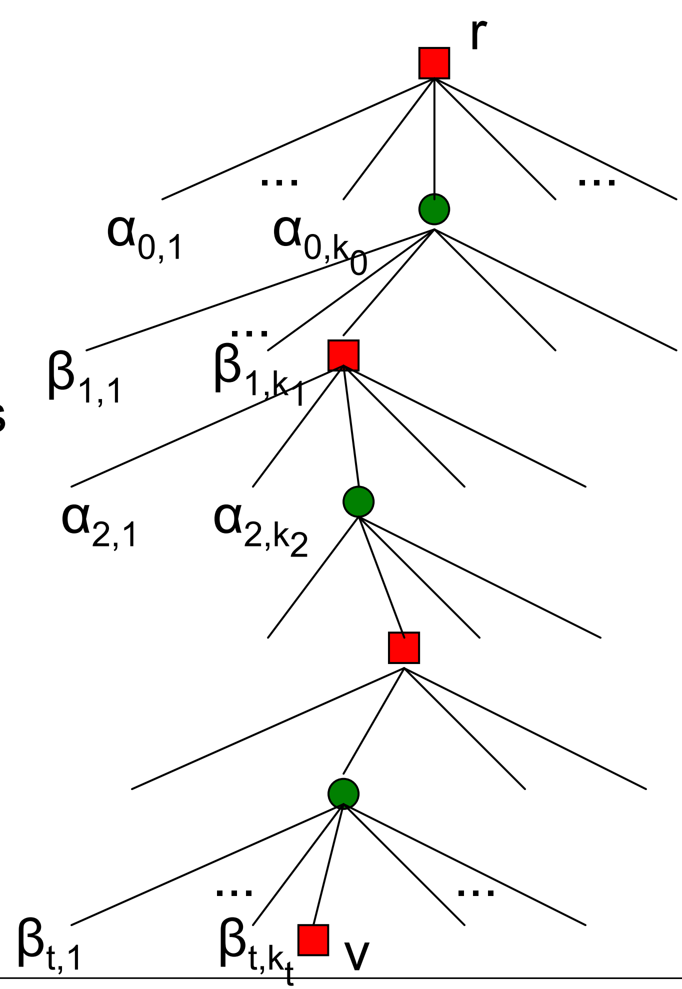
```
int alphabeta(Knoten v, int a, int b)
   if (v ist Blatt) return f(v);    // Blattbewertung
   for each child w of v do
      if (MAX-Spieler ist bei v am Zug)
         a = max(a, alphabeta(w, a, b));
         if (a >= b) return a;      // Beta-Cutoff
      else
         b = min(b, alphabeta(w, a, b));
         if (a >= b) return b;      // Alpha-Cutoff
   if (MAX-Spieler ist bei v am Zug) return a;
   else return b;
```

Alphabeta-Algorithmus

Der Alphabetaalgorithmus:

•Tiefensuche

- wenn die Suche an Knoten v angelangt, können wir uns vorstellen, dass
  - der durchsuchte Teil des Suchbaums „links“ von dem Weg von der Wurzel r zu v liegt, und
  - der noch nicht durchsuchte Teil „rechts“ des Weges liegt.



Alphabeta-Algorithmus

Es seien $\alpha_{i,j}$ ($\beta_{i,j}$) die Werte der linken Nachbarknoten von MIN- (bzw. MAX)- Knoten auf dem Weg von r nach v.

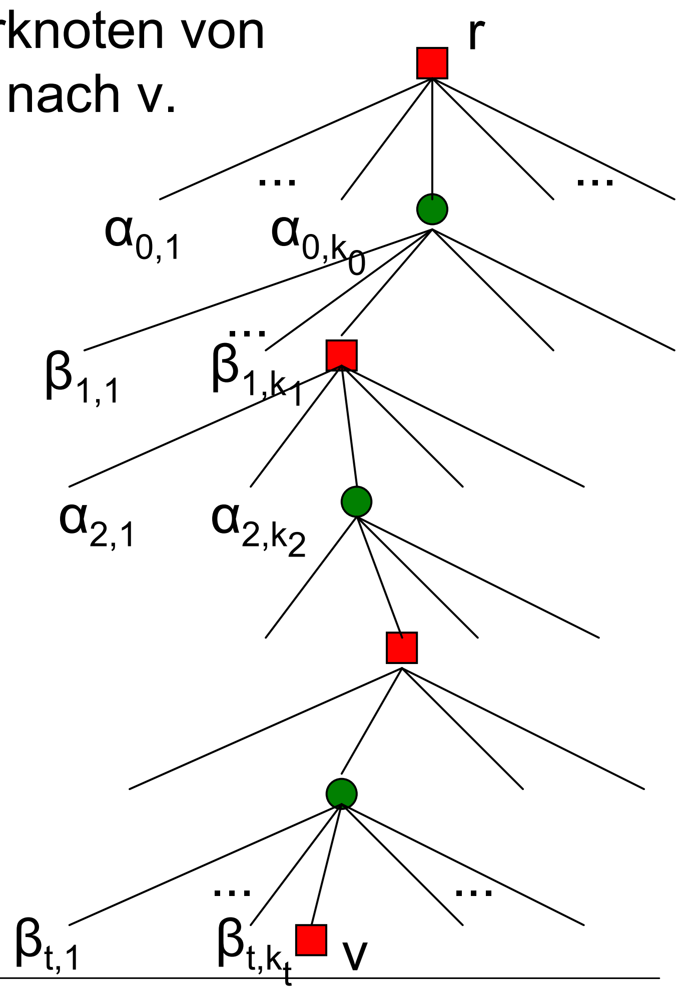
Sei

$\alpha(v) := \max_{i=0,\dots,t;\ i \text{ gerade}} \max_{j=1,\dots,k_i} \alpha_{i,j}$

und

$\beta(v) := \min_{i=0,\dots,t;\ i \text{ ungerade}} \min_{j=1,\dots,k_i} \beta_{i,j}$

Dann kann MAX bereits $\alpha(v)$ erreichen, MIN bereits $\beta(v)$. $F(v)$ ist also nur noch von Interesse, wenn $\alpha(v) < F(v) < \beta(v)$. Wurde Alphabeta($r,-\infty,\infty$) aufgerufen, wird der Alphabeta-Algorithmus für v mit Alphabeta($v,\alpha(v),\beta(v)$) aufgerufen.



Alphabeta-Algorithmus

Beobachtung: $\alpha(v)$ und $\beta(v)$ sind abhängig von dem bereits durchsuchten Teil des Suchbaums und damit von der Reihenfolger von Nachfolge-knoten.

Satz: Es sei $G=(V,E,f)$ ein Spielbaum, v ein Knoten aus V, $\alpha(v)$ und $\beta(v)$ die Schranken für v. Der Alphabeta-Algorithmus untersucht v genau dann, wenn $\alpha(v) < \beta(v)$.

Beweis: klar mit Überlegung von vorher und Cutoff-Bedingung im Algorithmus.

Alphabeta-Algorithmus, Negamax-Variante

```
int negamax(node v, int a, b)
{
  if (v ist Blatt) return f(v);      // f bewerte v jetzt aus Sicht des Ziehenden!
  erzeuge alle Nachfolger v_0 ... v_b-1 von v
  for (i = 0; i < b; i++) {
        a = max(a, -negamax(v_i, -b, -a))
        if (a ≥ b) return a;        // cutoff
  }
  return a;
}
```

Lemma: Sei $x = \text{negamax}(v,\alpha,\beta)$ und F beschreibe den Minimax-Wert aus Sicht des Spielers, der bei v am Zug ist. Dann gilt:

$$
\begin{array}{ll}
x \le \alpha & \text{falls } F(v) \le \alpha \\
x = F(v) & \text{falls } \alpha < F(v) < \beta \\
x \ge \beta & \text{falls } F(v) \ge \beta
\end{array}
$$

Beweis: Induktion über Höhe h des Spielbaums → Übung

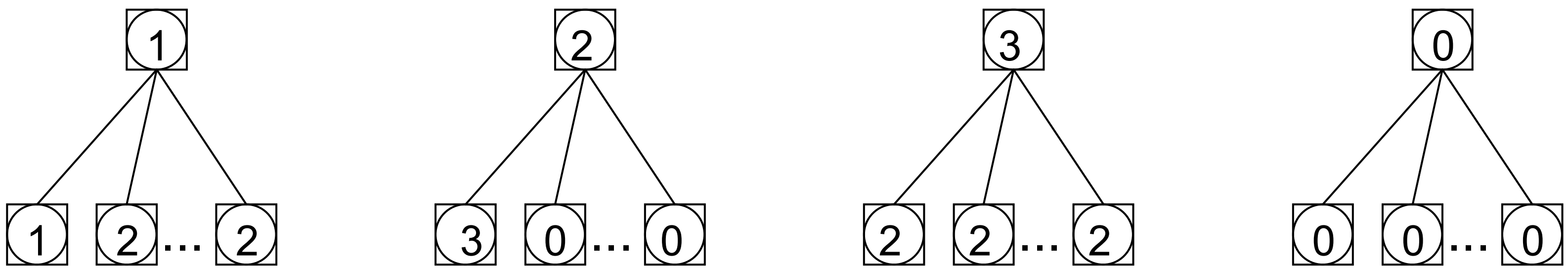
Alphabeta-Algorithmus, Aufwand

Sei G = (V,E,f) ein b/t-uniformer Spielbaum, d.h., alle inneren Knoten besitzen b Nachfolger und alle Blätter liegen in Tiefe t. Sei r seine Wurzel.

Beobachtung: Im schlimsten Fall besucht der Alphabeta-Algorithmus alle Knoten des Spielbaums.

Definition (Knotentyp, kritischer Knoten)

a) Die Abbildung ***Typ*** : V → {0,1,2,3} ist definiert durch Typ(r) = 1 und



b) ein Knoten v heißt ***kritisch***, wenn Typ(v) ≠ 0.

Alphabeta-Algorithmus, Aufwand

Satz: Es sei A ein beliebiger Algorithmus, der zu einem Spielbaum G = (V,E,f) mit Wurzel r dem Minimaxwert F(r) berechnet.

Dann kann man G durch Permutationen der Navchfolger der inneren Knoten so zu G‘ verändern, dass jeder Knoten, den der Alphabeta-Algorithmus auf G‘ besucht auch von A auf G besucht wird. Der Alphabeta-Algorithmus besucht gerade die kritischen Knoten.

Alphabeta-Algorithmus, Aufwand

Beweisskizze:

Für Knoten vom Typ 1,2,3 gelten folgende Aussagen, welche durch eine gemeinsame Induktion über die Tiefe von G bewiesen werden können:

A1: Sei v ein Knoten mit Typ(v) = 1. Dann gilt:

- A berechnet für v den Minimaxwert F(v).
- ist v ein Blatt, so wird von A f(v) berechnet,
- sonst:
  - muss A einen Nachfolger v_i von v gefunden haben, dessen untere Schranke den Wert F(v) besitzt. Ausserdem müssen für alle Nachfolger obere Schranken ≤ F(v) nachgewiesen worden sein. Insbesondere muss der Minimaxwert von v_i bestimmt worden sein. Permutiere die Nachfolger von v in G‘ nun so, dass v_i in G‘ der erste Nachfolger ist. In G weisen wir dem Knoten v_i den Typ 1 zu, allem anderen Nachfolgern von v den Typ 2.
- v wird mit negamax(v,-∞,∞) untersucht
- $Typ(v_i) = 1$ in G‘ $Typ(v_j) = 2$ für $j \neq i$ in G‘ und v_i wird in G‘ mit negamax(v_i,-∞,∞) untersucht, und alle anderen Nachfolger von v mit negamax(v_j,-∞,-F(v_i)).

  An den v_j kommt es zu Cutoffs, weil der erste Nachfolger von v in G‘ den größten Wert besitzt.

Alphabeta-Algorithmus, Aufwand

A2: Sei v ein Knoten mit Typ(v) = 2. Dann gilt:

- A berechnet für v zumindest eine untere Schranke $\beta > -\infty$.
- ist v ein Blatt, so wird von A f(v) berechnet,
- sonst:
  - muss A einen Nachfolger v_i von v gefunden haben, dessen untere Schranke $\beta > -\infty$ ist. Wir weisen v_i den Typ 3 zu, allen anderen Nachfolgern von v den Typ 0. Permutiere die Nachfolger von v in G‘ nun so, dass v_i in G‘ der erste Nachfolger ist.
- v wird mit negamax(v,$-\infty$,β) untersucht
- Typ(v_i) = 3 in G‘ Typ(v_j) = 0 für $j \neq i$ in G‘ und v_i wird in G‘ mit negamax(v_i,$-\beta$,∞) untersucht. negamax(v_i,$-\beta$,∞) liefert einen Wert $x \leq -\beta$ und es erfolgt ein Cutoff an v.

A3: Sei v ein Knoten mit Typ(v) = 3. Dann gilt:

- A berechnet für v eine obere Schranke für F(v). Er muss also alle Nachfolger betrachten. Alle Nachfolger bekommen den Typ 2.
- ist v ein Blatt, so wird von A f(v) berechnet,
- sonst: v wird in G‘ mit negamax(v,α,∞) untersucht, und Typ(v_j) = 2 für alle Nachfolger v_j von v. Die Nachfolger werden gesucht mit dem Aufruf negamax(v_j,$-\infty$,α).

Alphabeta-Algorithmus, Aufwand

Folgerung: Es sei G = (V,E,f) ein b/t-uniformer Spielbaum mit Wurzel r. Jeder Algorithmus zur Berechnung von F(r) besucht mindestens $b^{\lfloor t/2 \rfloor} + b^{\lceil t/2 \rceil} - 1$ Blätter von G.

Beweis:
Es sei $k_i(j)$ die Anzahl der Knoten v in Ebene j von G mit Typ(v) = i. Dann ist $k_1(0) = 1$, $k_2(0) = 0$ und $k_3(0) = 0$. Weiter ist

$$k_1(t) = k_1(t-1)$$
$$k_2(t) = (b-1) \cdot k_1(t-1) + b \cdot k_3(t-1)$$
$$k_3(t) = k_2(t-1)$$

Lösung des Systems ergibt:

$$k_1(t) = 1$$
$$k_2(t) = b^{\lceil t/2 \rceil} - 1$$
$$k_3(t) = b^{\lfloor t/2 \rfloor} - 1$$