EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

MATH. - NATURWISS. FAKULTÄT
FACHBEREICH INFORMATIK
KOGNITIVE SYSTEME · PROF. A. ZELL

Artificial Intelligence, Wintersemester 2012/2013

Übungsblatt 7

Abgabe: 13.12.2012, Besprechung: 20.12.2012

Aufgabe 1 Minimax [7 Punkte]

Gegeben sei ein Haufen mit drei und ein Haufen mit zwei Streichhölzern. Die beiden Spieler MAX und MIN ziehen abwechselnd jeweils ein oder mehrere Hölzer **von einem der beiden** Haufen. Es **verliert** derjenige, der das letzte Streichholz nimmt. MAX hat den ersten Zug.

(a) Bauen Sie den gesamten Suchraum für das Spiel auf. *Hinweis*: Identische Zustände derselben Tiefe dürfen zusammengefasst und somit ein gerichteter, azyklischer Graph gezeichnet werden.

(b) Bewerten Sie Gewinnstellungen für MAX mit $+1$ und Gewinnstellungen für MIN mit -1. Berechnen Sie die **minimax**-Werte für sämtliche Knoten.

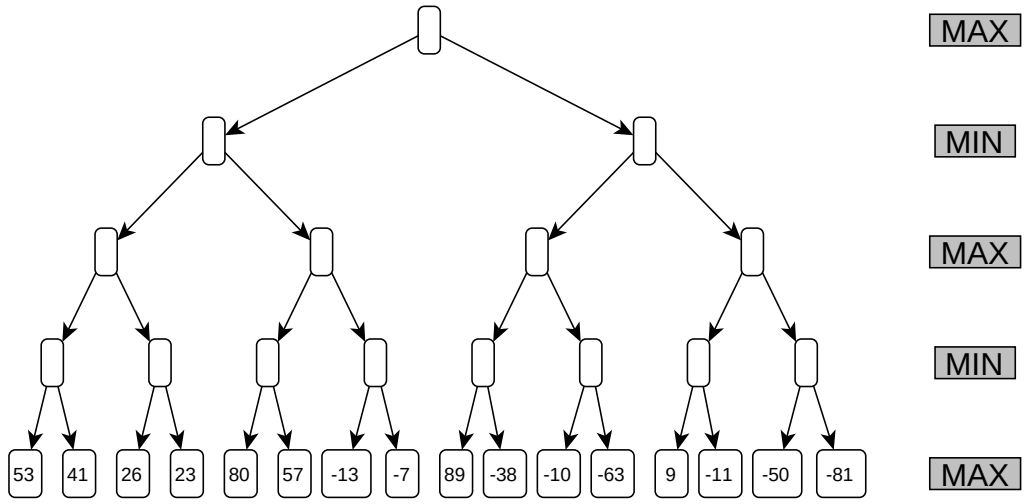
(c) Gibt es eine sichere Gewinnstrategie für einen der beiden? Falls ja, welche? Wenn nein, warum nicht?

Aufgabe 2 Alpha-Beta Pruning [7 Punkte]

Während einer Spielsituation, in der MAX am Zug ist, ergebe eine Vorausschau um zwei Züge (vier Halbzüge) die unten dargestellten Bewertungen der Nachfolgesituation.

(a) Geben Sie den **minimax**-Wert für jeden Knoten an. Welchen Zug wählt MAX optimalerweise? Welchen Zustand an den Blättern des Baumes erwartet MAX zu erreichen?

(b) Wenden Sie die **Alpha-Beta-Pruning**-Strategie auf diesen Baum an. In welcher Folge werden die Werte berechnet? Welche Teile des Suchbaums fallen weg?



Aufgabe 3 Minimax mit Alpha-Beta Pruning in Java [6 Punkte]

Laden Sie die Datei `uebung07.zip` von der Übungs-Webseite herunter. Darin finden Sie das Gerüst für einen Minimax-Algorithmus, mit und ohne Alpha-Beta Pruning. Außerdem finden Sie dort Implementierungen der Spiele *Vier Gewinnt* (engl. *Connect Four*) und *Tonylanche*, die beide die Schnittstellen (*Interfaces*) `Game` und `State` implementieren. Diese definieren Methoden, die den in der Vorlesung vorgestellten Methoden entsprechen. `getUtility(State s, int p)` z. B. entspricht der Funktion $UTILITY(s, p)$.

Regeln für Tonylanche

Zwei Spieler ziehen abwechselnd Steine auf dem Spielfeld. Der Spieler wählt einen Stein, den er nach links bewegen möchte. Der Stein darf so viele Felder wie möglich bewegt werden, mindestens aber ein Feld, und er kann nicht über einen anderen Stein oder den Spielfeldrand bewegt werden. Es gewinnt der Spieler, der den letzten Zug vollzieht.

Beispiel (Buchstaben sind Steine):

				A	B		
			C				

(a) Vervollständigen Sie die Klasse `GeneralMinMaxPlayer` an den mit TODO markierten Stellen, so dass diese den Minimax-Algorithmus jeweils mit und ohne Alpha-Beta Pruning implementiert. Benutzen Sie dabei keine Referenzen auf die konkreten Implementierungen der Spiele, sondern nur die allgemeinen Interfaces `Game` und `State`.

(b) Lassen Sie dann zwei Instanzen Ihres Minimax-Algorithmus für beide Spiele mit und ohne Alpha-Beta Pruning gegeneinander spielen. Nutzen Sie für die Startzustände die Methoden `getInitialStateEasy()` und `getInitialStateMedium()`
Dokumentieren Sie die Ergebnisse, insbesondere die Anzahl der generierten Knoten des Suchbaums. Diskutieren Sie Verbesserungsmöglichkeiten.
Hinweis: Sollte Ihr Algorithmus länger als fünf Minuten für ein Spiel benötigen, dürfen Sie das Programm abbrechen.

Senden Sie Ihre kommentierte Implementierung sowie Ihre Ergebnisse an andreas.draeger@uni-tuebingen.de und florian.mittag@uni-tuebingen.de mit dem Betreff „Abgabe KI-Uebung 07“.