Informatik II (D-ITET)
FS 2009
Prof. Dr. F. Mattern
Philipp Bolliger (bolligph@inf.ethz.ch)

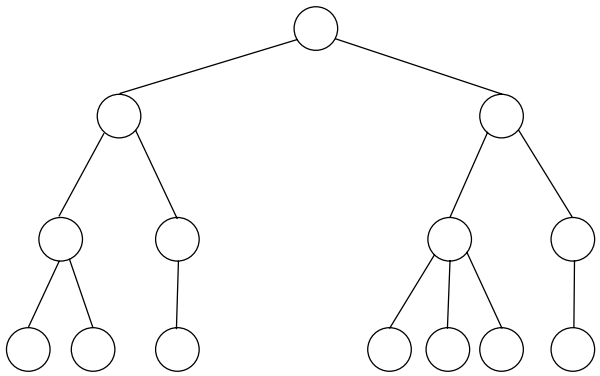
ETH Zürich
Institut für Pervasive Computing
Verteilte Systeme
http://www.vs.inf.ethz.ch

Übungsserie Nr. 9

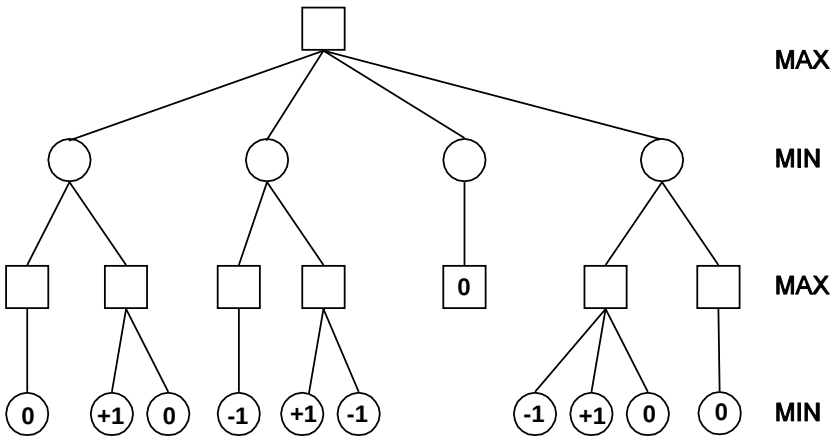
Ausgabe: 29. April 2009
Abgabe: 6./7. Mai 2009

1. Aufgabe: (8 Punkte) Spieltheorie

(1a) *(1 Punkt)* Wie gross ist die Suchtiefe dieses Baums? Wie gross ist seine Höhe? (*Ein Baum mit nur einem Knoten habe die Höhe 1.*)

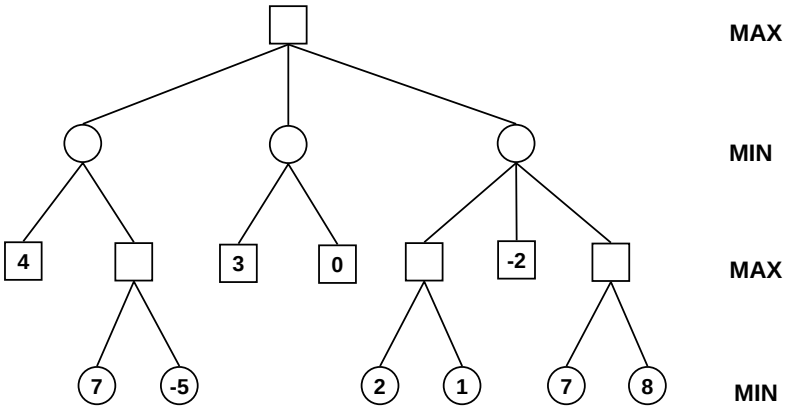


(1b) *(2 Punkte)* Wenden Sie die Minimax-Regel auf den folgenden Spielbaum an und markieren Sie dabei alle Knoten mit den entsprechenden abgeleiteten Werten (vgl. Skript Folie 301). Finden Sie damit den besten Zug in der aktuellen Position für den Spieler MAX.



(1c) *(2 Punkte)* Betrachten Sie den vorherigen Spielbaum und geben Sie eine optimale Strategie für den Spieler MAX an (vgl. Skript Folie 289–297).

(1d) *(3 Punkte)* Berechnen Sie den Wert der Wurzel des folgenden Spielbaums mit Hilfe der α-β-Methode (vgl. Skript Folie 311 ff.). Markieren Sie die Stellen mit α und β, wo Sie einen entsprechenden Schnitt gemacht haben. Markieren Sie auch die Knoten, die für die Berechnung keine Rolle spielen.



2. Aufgabe: (10 Punkte) Reversi [Teil 3]

Bei unserem Reversi-Turnier am Ende des Semesters wird jedem Computerspieler nur eine begrenzte Zeit pro Spielzug zur Verfügung stehen.

(2a) *(5 Punkte)* Implementieren Sie eine Methode, die den Spielbaum mit Minimax (oder Negamax) maximal bis zur Tiefe d auswertet.

(2b) *(5 Punkte)* Vor Ablauf von `timeLimit` Millisekunden soll Ihre Methode `nextMove()` einen gültigen Zug zurückgegeben. (Der Parameter `timeLimit` wird beim Aufruf der Methode `init()` an Ihr Programm übergeben.) Erweitern Sie Ihr Programm so, dass es periodisch die verstrichene Zeit abfragt. Die Zeile

```
long currentTime = System.currentTimeMillis();
```

liefert die Anzahl der seit dem 1. Januar 1970, 00:00:00 (GMT) vergangenen Millisekunden zurück.

3. Aufgabe: (3 Punkte) Sortieren mit Suchbäumen

(3a) *(1 Punkt)* Beschreiben Sie kurz, wie man binäre Suchbäume zum Sortieren benutzen kann.

(3b) *(2 Punkte)* Ein binärer Suchbaum soll durch das Einfügen von Elementen aus einer Liste neu aufgebaut werden. Überlegen Sie sich, welche der folgenden Aussagen bezüglich der Anzahl der dabei notwendigen Vergleichsoperationen richtig sind.

Im günstigsten Fall sind die Werte der Liste

a) aufsteigend vorsortiert,

b) absteigend vorsortiert,

c) gut durchmischt.

Im schlechtesten Fall sind die Werte der Liste

d) aufsteigend vorsortiert,

e) absteigend vorsortiert,

f) gut durchmischt.

Hinweis: Zwei binäre Suchbäume können unterschiedlich aussehen, obwohl sie die gleichen Schlüsselattribute enthalten. Ausschlaggebend für die Gestalt des Suchbaumes ist die Reihenfolge der Einfügungen.

4. Aufgabe: (4 Punkte, freiwillige Aufgabe) Java Collections-Framework

Im Java-Package `java.util` finden Sie eine Reihe von Klassen zur Verwaltung einer dynamischen Anzahl von Objekten in einer Liste (wie Sie es im Rahmen der `ObjectList` der Übungsserie 6 selbst implementiert haben). Schauen Sie sich in der Java-API dazu die Dokumentation des Interfaces `List` an [1].

(4a) *(2 Punkte)* Implementieren Sie (in einer eigenen Klasse) eine *rekursive* Methode

```
public List reverse(List list);
```

die das Argument in umgekehrter Reihenfolge zurückgibt. Verzichten Sie auf die Verwendung von Schleifen.

Tipp: Sie könnten pro Rekursionsschritt den Rest der Liste (alles ausser dem ersten Element) umkehren und danach das erste Element an geeigneter Stelle wieder einfügen. Benutzen Sie dabei ***nur*** *Methoden des Interfaces* `List`*, damit die Methode mit jeder Art von Liste arbeiten kann.*

(4b) *(2 Punkte)* Testen Sie die Methode in `main` mit einer geeigneten Liste. Beachten Sie, dass Sie dort eine spezifische Liste instanziieren müssen. Verschaffen Sie sich einen Überblick über die verschiedenen Java-Klassen, die die Funktionalität einer Liste bereitstellen (also das Interface `List` implementieren, insbesondere `ArrayList`, `LinkedList` und `Vector`).

Denkaufgabe: Geben Sie eine Empfehlung ab, welche Klasse für das rekursive Umkehren wahrscheinlich am geeignetsten ist (d.h., die in `reverse` verwendeten Operationen am effizientesten implementieren kann). Ändern Sie `main` entsprechend.

Summe: 21(+4) Punkte

[1] http://java.sun.com/j2se/1.5.0/docs/api/java/util/List.html