Informatik II (D-ITET)
FS 2010
Prof. Dr. F. Mattern
A. Bernauer (bernauer@inf.ethz.ch)

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

Übungsserie Nr. 9

Ausgabe: 5. Mai 2010
Abgabe: 12. Mai 2010

Hinweise

Für diese Serie benötigen Sie die Archive
http://www.vs.inf.ethz.ch/edu/I2/downloads/u9.zip
und http://www.vs.inf.ethz.ch/edu/I2/downloads/reversi.jar.

1. Aufgabe: (12 Punkte) Rucksackproblem und Backtracking

Ein Dieb ist in ein Haus eingebrochen und steht nun vor dem Problem, unter den K vorhandenen Gegenständen $x_1, \ldots, x_K$ eine Auswahl treffen zu müssen, so dass das zulässige Gesamtgewicht G seiner Tasche nicht überschritten wird. Gleichzeitig soll der Wert aller eingepackten Objekte möglichst gross sein. Der Dieb kennt von jedem Gegenstand dessen Wert $w_i \geq 0$ sowie dessen Gewicht $g_i \geq 0$. Formal ist also eine Selektion $b_1, \ldots, b_K$ mit $b_i \in 0, 1$ gesucht, für die gilt:

$$\sum_{i=1}^{K} b_i w_i \text{ ist maximal, und } \sum_{i=1}^{K} b_i g_i \leq G$$

(1a) *(2 Punkte)* Gibt es immer genau eine optimale Lösung? Beweisen Sie Ihre Antwort.

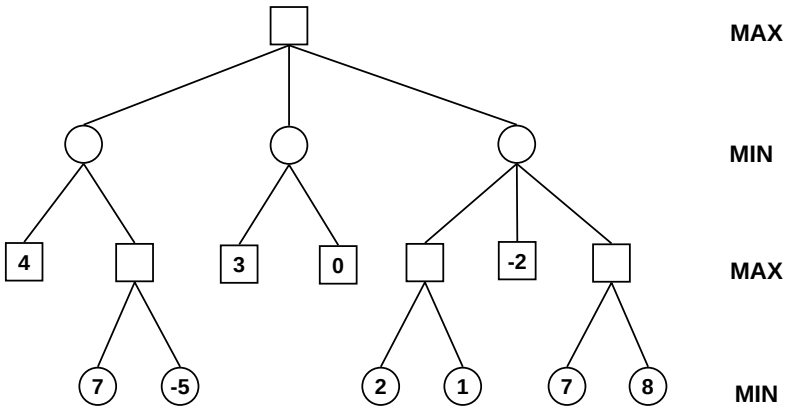
(1b) *(4 Punkte)* Implementieren Sie die Schnittstelle *IRucksack* mit einem Algorithmus, der alle Möglichkeiten ausprobiert und die beste davon zurück liefert. Passen Sie zum Testen die Fabrikmethode *RucksackFactor.create* an.

(1c) *(4 Punkte)* Implementieren Sie die Schnittstelle *IRucksack* mit einem Backtracking-Algorithmus. Passen Sie zum Testen die Fabrikmethode erneut an.

(1d) *(2 Punkte)* Der Test *complex* misst die Zeit, die Ihr Algorithmus braucht, um eine Lösung zu finden. Vergleichen Sie die Werte für Ihre Algorithmen aus Aufgabe b und c. Was fällt Ihnen auf? Wie erklären Sie sich das?

2. Aufgabe: (8 Punkte) Spieltheorie

Die Aufgaben in diesem Teil beziehen sich alle auf den folgenden Spielbaum:



(2a) *(1 Punkt)* Wie gross ist die Suchtiefe dieses Baums? Wie gross ist seine Höhe? (*Ein Baum mit nur einem Knoten habe die Höhe 1.*)

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

(2c) *(2 Punkte)* Geben Sie für den Spielbaum eine optimale *Strategie* für den Spieler MAX an.

(2d) *(3 Punkte)* Berechnen Sie den Wert der Wurzel des Spielbaums mit Hilfe der α-β-Methode. Versehen Sie dabei jeden Ast mit den aktuellen Werten für α und β. Markieren Sie ausserdem die Stellen, an denen Sie einen entsprechenden Schnitt gemacht haben.

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

(3a) *(5 Punkte)* Implementieren Sie einen Reversi-Spieler, der eine Min-max-Analyse der Spielsituation durchführt. Machen Sie dabei die maximale Suchtiefe konfigurierbar und verwenden Sie als Bewertungsfunktion wieder die Differenz der Spielsteine.

(3b) *(5 Punkte)* Im Turnier hat jeder Spieler nur eine begrenzte Zeit zur Verfügung. Wie lange die ist, wird dem Spieler durch die Methode *initialize* mitgeteilt.

Passen Sie Ihren Reversi-Spieler so an, dass er die Min-max-Analyse mit ständig wachsender Suchtiefe so lange immer wieder durchführt, bis ihm die Zeit ausgeht und dann das Ergebnis der letzten vollständigen Analyse zurück gibt.

Hinweis: Die Methode *System.currentTimeMillis* liefert einem die aktuelle Systemzeit in Millisekunden seit dem 1. Januar 1970, 00:00:00 (GMT) zurück.

Hinweis: Der *LogPlayer* kann ein vorheriges Spiel nachahmen, in dem er die Züge aus einem Logfile liest. Das kann hilfreich sein, um Spielsituationen wiederherzustellen, in denen Ihr Spieler sich falsch verhalten hat.

(3c) *(freiwillig)* Implementieren Sie eine bessere Bewertungsfunktion. Orientieren Sie sich dabei an der auf der Reversi-Seite verlinkten Literatur.