Informatik II (D-ITET)
Frühjahrssemester 2013
Prof. F. Mattern
S. Mayer (simon.mayer@inf.ethz.ch)

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

# Übungsserie Nr. 11

Ausgabe: 15. Mai 2013
Abgabe: 22. Mai 2013

**Hinweise**

Für diese Serie benötigen Sie das Archiv
http://vs.inf.ethz.ch/edu/FS2013/I2/downloads/u11.zip.

Das Reversi-Turnier findet am 29. Mai ab 14:15 Uhr im StuZ (CABinett) statt. Wer mit einem eigenen Spieler antreten möchte, beachtet bitte die Teilnahme- und Abgabebedingungen auf der Reversi-Webseite[a] (**Abgabefrist: 22. Mai 2012**). Dort finden Sie auch die Preise für die bestplatzierten Teams. Auch Studenten, die keinen Spieler einreichen, sind herzlich eingeladen, dabei zu sein - zusammen mit dem AMIV sorgen wir für Getränke und Snacks.

[a] http://vs.inf.ethz.ch/edu/FS2013/I2/reversi

## 1. Aufgabe: (5 Punkte) Sortieren mit Suchbäumen

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

(1b) *(2 Punkte)* Ein binärer Suchbaum soll durch das Einfügen von Elementen aus einer Liste aufgebaut werden. Welche der folgenden Anfangsbedinungen sind dabei besonders günstig bzw. ungünstig? Begründen Sie!

a) aufsteigend vorsortiert

b) absteigend vorsortiert

c) gut durchmischt

(1c) *(2 Punkte)* Geben Sie die Laufzeitkomplexität einer Sortierung mit binären Suchbäumen für den besten, den durchschnittlichen und den schlechtesten Fall in O-Notation an.

## 2. Aufgabe: (6 Punkte) Komplexitätsanalyse und O-Notation

(2a) *(6 Punkte)* Geben Sie für jedes der folgenden sechs Codefragmente die Laufzeitkomplexität in $O$-Notation an. Nehmen Sie dabei an, dass jede arithmetische Operation den Aufwand 1 hat. Nehmen Sie ausserdem an, dass es in jedem Fragment eine *int*-Variable namens *a* gibt.

```
// Fragment 1
   for (int i=1; i<n; i++) a++;

// Fragment 2
   for (int i=0; i<2*n; i++) a++;
   for (int j=0; j<n; j++) a++;

// Fragment 3
   for (int i=0; i<n; i++)
      for (int j=0; j<n; j++)
           a++;
```

```
// Fragment 4
   for (int i=0; i<n; i++)
      for (int j=0; j<i; j++)
         a++;

// Fragment 5
   while(n >= 1) n = n/2;

// Fragment 6
   for (int i=0; i<n; i++)
      for (int j=0; j<n*n; j++)
         for (int k=0; k<j; k++)
            a++;
```

## 3. Aufgabe: (4 Punkte) Komplexität

(3a) *(4 Punkte)* Gegeben seien 4 verschiedene Algorithmen mit den unten angeführten Laufzeitkomplexitäten in Anzahl der benötigten Rechenoperationen. Es ist weiters bekannt, dass auf einem gegebenen System in einer fest vorgegebenen Zeit je eine maximale Eingabegrösse verarbeitet werden kann. Nun wird dieses System durch ein neues System ersetzt, welches dreimal so schnell arbeitet. Welche Eingabegrössen können mit diesem System in der vorgegebenen Zeit maximal bearbeitet werden?

| Laufzeit | max. Grösse alt | max. Grösse neu |
|---|---|---|
| $O(n)$ | $M_1$ | |
| $O(n^2)$ | $M_2$ | |
| $O(2^n)$ | $M_3$ | |
| $O(\log n)$ | $M_4$ | |

## 4. Aufgabe: (8 Punkte) Ein Springer auf dem Schachbrett

Im Schach bewegt sich ein Springer entweder um zwei Felder horizontal und eines vertikal oder um ein Feld horizontal und zwei vertikal. Dieses Sprungmuster führt zu ein paar interessanten Fragen, denen Sie hier nachgehen sollen.

(4a) *(4 Punkte)* Implementieren Sie die Methode *getReachableSet* der Schnittstelle *IKnight*, die die Menge der Felder auf dem Schachbrett zurückgibt, die der Springer ausgehend von einer Startposition in einer gegebenen maximalen Anzahl von Schritten erreichen kann.

(4b) *(4 Punkte)* Implementieren Sie die Methode *findCompletePath* der Schnittstelle *IKnight*, die zu einer gegebenen Startposition einen Pfad über das Schachbrett zurückliefert, bei dem alle Felder genau einmal besucht werden.

*Hinweis: die Laufzeit hängt* **sehr stark** *von der Reihenfolge ab, in der die acht möglichen Züge durchprobiert werden. Der Unterschied kann sich dabei im Bereich von mehreren Minuten bewegen. Daher empfehlen wir Ihnen, für diese Aufgabe die folgende Sprungreihenfolge zu verwenden:*