Prof. Dr. Gerhard Reinelt
Hanna Seitz
Institut für Informatik
Universität Heidelberg

# Algorithmen & Datenstrukturen

2. Übungsblatt SS 07
Abgabetermin: 09.05.2007

## Aufgabe 5

Lösen Sie die folgenden Rekursionsgleichungen.

**a)** $T(n) = T(\frac{2n}{3}) + 1$

**b)** $T(n) = 4 \cdot T(\frac{n}{2}) + n$

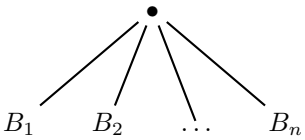
**c)** $T(n) = 4 \cdot T(\frac{n}{2}) + n^2$

**d)** $T(n) = 4 \cdot T(\frac{n}{2}) + n^3$

## Aufgabe 6

Ein **Baum** lässt sich wie folgt rekursiv definieren:

- Ein einzelner **Knoten** • ist ein Baum.
- Wenn $B_1, B_2, \ldots, B_n, n \geq 1$, Bäume sind, dann ist auch




ein Baum. Hier hängen also $n$ Bäume an einem Knoten. Die Verbindungslinien heissen **Kanten**. Die Unterstrukturen $B_1, B_2, \ldots, B_n$ nennt man **Teilbäume** oder **Unterbäume**.

Den eindeutig bestimmten Knoten, der keinen übergeordneten Knoten hat, nennt man den **Wurzelknoten** oder die **Wurzel** des Baumes. Knoten, die keine untergeordneten Knoten haben, nennt man **Blätter**.

Beschreiben Sie einen abstrakten Datentyp **Baum** mit den Operationen `createTree`, `insertTree`, `deleteTree` und skizzieren Sie eine Implementierung. Geben Sie die Worst-Case-Laufzeit für die einzelnen Operationen an.

## Aufgabe 7

Gegeben seien $n$ Städte $1, \ldots, n$ sowie die Distanzen $c_{ij}$ zwischen je zwei Städten $i$ und $j$. Das **Traveling Salesman Problem (TSP)** besteht darin, eine kürzeste Rundreise zu finden, bei der jede Stadt genau einmal besucht wird und welche in der zuerst besuchten Stadt endet.

Der folgende Algorithmus bestimmt eine Näherungslösung für das TSP.

**(1)** Wähle eine beliebige Stadt $j$, setze $l = j$ und $W = \{1, 2, \ldots, n\} \setminus \{j\}$.

**(2)** Solange $W \neq \emptyset$:

  **(2.1)** Sei $j \in W$, so dass $c_{lj} = \min\{c_{li} \mid i \in W\}$.
  **(2.2)** Verbinde Stadt $l$ mit Stadt $j$, setze $W = W \setminus \{j\}$ und $l = j$.

**(3)** Verbinde Stadt $l$ mit der zuerst besuchten Stadt aus Schritt (1) und vervollständige somit die Rundreise.

Nun zu Ihrer Aufgabe.

**a)** Skizzieren Sie eine Implementierung des Algorithmus in C (oder C++).

**b)** Analysieren Sie die Laufzeit Ihrer Implementierung in Abhängigkeit von $n$.