

## Datenstrukturen und Algorithmen – Informatik I

### 10. Übung

Abgabe der Lösungen: 12. Juli 2004

---

**Hinweis:** Die Implementierungsaufgaben sind in Java oder C/C++ zu lösen. Der Quellcode sowie ein dokumentierter Beispiellauf sind beim Betreuer der Kleingruppe abzugeben.

#### Aufgabe 43: *B*-Baum

**2 Punkte**

Beweisen oder widerlegen Sie die folgende Aussage: Bei gegebenen Schlüsselwerten ist die Anzahl der Knoten des entsprechenden *B*-Baumes eindeutig.

#### Aufgabe 44: Implementierung: Prioritätswarteschlangen

**6 Punkte**

In Übung 7, Aufgabe 33 wurden die Realisierungen von Prioritätswarteschlangen unter Verwendung verschiedener Datenstrukturen betrachtet. Implementieren Sie nun Prioritätswarteschlangen unter Verwendung eines Heaps und realisieren Sie die folgenden Operationen:

**Initialize**( $S$ ) initialisieren von  $S$  mit der leeren Menge  $\emptyset$ .

**Insert**( $x, S$ ) fügt Element  $x$  in die Warteschlange ein.

**ReadMin**( $S$ ) gibt das Element mit dem kleinsten Schlüssel zurück (ohne es zu entfernen).

**DeleteMin**( $S$ ) entfernt das Element mit dem kleinsten Schlüssel.

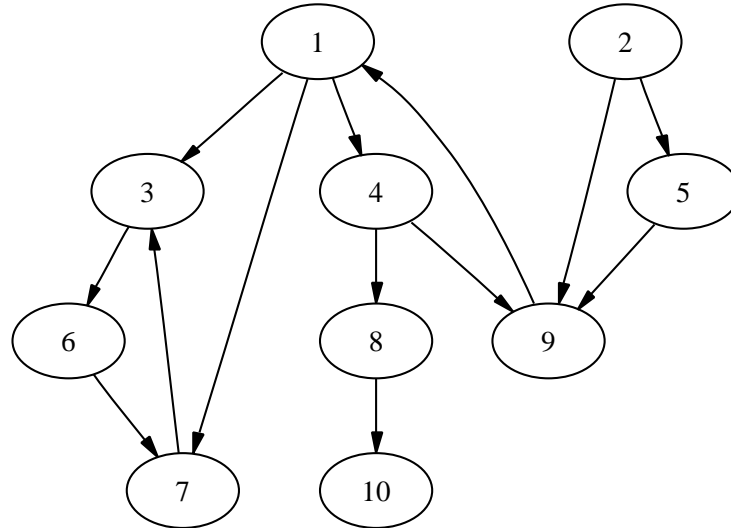
**ReplaceMin**( $x, S$ ) entfernt das Element mit dem kleinsten Schlüssel aus  $S$  und fügt  $x$  an die richtige Position in  $S$  ein.

$S$  ist hier die nach Priorität geordnete Schlüsselmenge. Alle Operationen sollen die Eigenschaft der Prioritätswarteschlange beibehalten, d.h. die Schlüssel müssen ggfs. nach dem Löschen bzw. Einfügen an die richtige Position bewegt werden.

#### Aufgabe 45: Schnittgraphen

**4 Punkte**

Seien  $G_1 = (V_1, E_1)$  und  $G_2 = (V_2, E_2)$  zwei Graphen. Der Schnitt von  $G_1$  und  $G_2$  ist folgendermaßen definiert:  $G_1 \cap G_2 = (V, (E_1 \cup E_2) \cap (V \times V))$  mit  $V = V_1 \cap V_2$ . Geben Sie einen Algorithmus für Graphen in Adjazenzlistendarstellung an, der  $G_1 \cap G_2$  berechnet. Dabei können Sie annehmen, daß die Adjazenzlisten nach der Knotenbeschriftung sortiert sind. Geben Sie die Laufzeitkomplexität dieses Algorithmus an.

**Aufgabe 46: Darstellungen und Durchläufe von Graphen****2+2+2 Punkte**Sei  $G_0 = (\{1, \dots, 10\}, E)$  der folgende Graph:

1. Geben Sie Adjazenzmatrix und Adjazenzliste für diesen Graphen an.
2. Geben Sie für den Graphen  $G_0$  den Verlauf der Tiefensuche beginnend bei Knoten 1 an (Nachfolgerknoten werden in aufsteigender Reihenfolge besucht).
3. Führen Sie für den Graphen  $G_0$  die Breitensuche beginnend bei Knoten 1 durch. Geben Sie die Reihenfolge der Knotenbesuche an und heben Sie die entstehenden Baumkanten hervor.

**Aufgabe 47: Textformatierung mit dynamischer Programmierung****6 Punkte**

Wir betrachten das Problem, einen Text in schönem Format auf einem Drucker auszugeben. Eingabe sei eine Sequenz von  $N$  Wörtern  $w_1, \dots, w_N$  der Längen  $l_1, \dots, l_N$  in Zeichen. Dieser Text soll in Zeilen, die höchstens  $M$  Zeichen enthalten können, ausgegeben werden. Eine Zeile, die die Wörter  $w_i$  bis  $w_j$  enthält, die jeweils durch ein Leerzeichen getrennt werden, enthält zusätzlich  $M - j + i - \sum_{k=i}^j l_k$  Leerzeichen. Ziel ist, die Summe über die dritten Potenzen der Anzahl der zusätzlichen Leerzeichen aller (außer der letzten) Zeilen zu minimieren. Es wird davon ausgegangen, daß alle Zeichen die gleiche Breite haben.

Also muß

$$R(Z, (i_1, \dots, i_Z), (j_1, \dots, j_Z)) = \sum_{z=1}^{Z-1} \left( M - j_z + i_z - \sum_{k=i_z}^{j_z} l_k \right)^3$$

über seine Parameter  $Z, (i_1, \dots, i_Z), (j_1, \dots, j_Z)$  minimiert werden. Dabei ist  $Z$  die (unbekannte) Anzahl der Zeilen,  $i_z$  das erste Wort der Zeile  $z$  und  $j_z$  das letzte Wort der Zeile  $z$ .

1. Geben Sie eine geeignete Rekursionsgleichung für die dynamische Programmierung an.
2. Geben Sie einen geeigneten Algorithmus in Pseudocode an.

**Hinweis:** Diese Übung ist eine *Auswahlübung*, d.h. von den 24 Punkten dieser Übung werden nur 20 bei der Berechnung der zu erreichenden Gesamtpunktzahl berücksichtigt. Wenn eine Gruppe mehr als 20 Punkte erreicht, so sind dies Zusatzpunkte, die normal gewertet werden.