Algorithmen und Datenstrukturen (PI.ADS.AD.VO)	schriftliche Einzelpruefung	03.07.2007		1

Aufgabe 1 [20]

Addieren Sie die letzten beiden Ziffern Ihrer Matrikelnummer mit 251470. Nehmen Sie an, die so entstandene Ziffernfolge würde in dieser Reihenfolge in einem zu sortierenden Feld einzelner Ziffern stehen.

a. [10] Sortieren Sie die Ziffernfolge aufsteigend mittels Heapsort. Geben Sie den Zustand des Feldes und den entsprechenden Baum nach jedem Schritt des Sortieralgorithmus (ein Schritt entspricht dabei dem Vertauschen zweier Ziffern) an.

b. [5] Sortieren Sie die so erhaltene Ziffernfolge (also die sortierte, *nicht* die ursprüngliche Folge) mit den Verfahren Insertion Sort und Quicksort.

   Geben Sie das durchschittliche Laufzeitverhalten der beiden Verfahren in O-Notation an.

c. [5] Wie viele Vergleichsoperationen haben Sie für Insertion Sort, bzw. Quicksort benötigt? Entspricht das dem durchschnittlich zu erwartenden Verhalten? Welcher Algorithmus erzielt die bessere Laufzeitordnung? Kennen Sie einen Algorithmus, der für diesen Fall eine noch bessere Laufzeitordnung aufweisen würde?

Aufgabe 2 [20]

Gegeben ist die folgende symmetrische Kostenmatrix, die die Gewichte der Kanten in einem ungerichteten Graphen definiert:

$$\begin{pmatrix} 0 & 7 & 5 & z1 & z2 \\ & 0 & z3 & z4 & 0 \\ & \cdots & z5 & 8 & 9 \\ & & & 1 & z6 \\ & & & & 0 \end{pmatrix}$$

Die Einträge in der Matrix sind so zu interpretieren, dass 0 keine Kante bedeutet, und jede andere Zahl das Gewicht der entsprechenden Kante festlegt.

Ersetzen Sie in der Matrix die Gewichte z1 bis z6 durch Werte, die Sie aus Ihrer Matrikelnummer wie folgt ermitteln: zi ergibt sich aus der i-ten Stelle der Matrikelnummer (von rechts beginnend nummeriert) plus 1. Für die Matrikelnummer 1234567 wäre z2 beispielsweise 7 (=6+1).

a. [10] Skizzieren Sie den so definierten Graphen und ermitteln Sie mit Hilfe des Algorithmus von Prim einen minimal spannenden Baum für diesen Graphen. Geben Sie den Zustand des Graphen nach jedem Hinzufügen einer Kante an.

b. [10] Versehen Sie die Knoten des Graphen mit Namen (z.B. indem Sie sie durchnummerieren) und geben Sie die Reihenfolgen an, in der die Knoten besucht werden, wenn Sie den erhaltenen spannenden Teilbaum (ausgehend vom Knoten, dessen Verbindungen in der ersten Zeile der Matrix eingetragen sind) mit bfs bzw. mit dfs traversieren.

Aufgabe 3 [20]

Hashverfahren

a. [5] Beschreiben Sie die grundlegende Idee von statischen Hashverfahren. Geben Sie die Aufwandsabschätzung für Einfügen, Löschen und Suchen eines Elementes in O-Notation an. Welche Verfahren haben Sie in der Vorlesung kennengelernt und wie unterscheiden sich diese?

b. [15] Was sind dynamische Hashverfahren? Was ist der grundlegende Unterschied zu den statischen Verfahren? Welche Verfahren kennen Sie aus der Vorlesung? Beschreiben Sie das Verhalten eines dieser Verfahren beim Einfügen anhand eines konkreten Beispiels (dabei sind die Parameter und die einzufügenden Zahlen so zu wählen, dass die Hashtabelle mindestens einmal vergrößert wird).

Aufgabe 4 [20]

Bäume

a. [10] Beschreiben Sie kurz die Datenstrukturen binärer Suchbaum, 2-3-4 Baum und B+-Baum. Was sind die Gemeinsamkeiten und wo liegen die Unterschiede? Beschreiben Sie jeweils kurz die Operationen Einfügen und Löschen in den verschiedenen Bäumen.

b. [10] Finden Sie ein Beispiel mit 5 einzutragenden Werten, sodass der binäre Suchbaum entartet (sein schlechtestes Laufzeitverhalten zeigt). Tragen Sie die selben Werte in der gleichen Reihenfolge in einen 2-3-4 Baum und in einen B+-Baum der Ordnung 2 ein. Ist die von Ihnen gewählte Folge von Werten für den 2-3-4 Baum bzw. den B+-Baum auch ungünstig (d.h. wird nicht das durchschnittliche Laufzeitverhalten erreicht)?

Aufgabe 5 [20]

Gegeben sind folgende Funktionen:

```
double* f(int n, double* a, double* b) {
  double* res=new double[n*n];

  for (int i=0; i<n*n; ++i)
    res[i]=0;

  for (int i=0; i<n; ++i)
   for (int j=0; j<n; ++j)
     for (int k=0; k<n; ++k)
       res[i*n+j]+=a[i*n+k]*b[k*n+j];

  return res;
}

int g(int m) {
  if (m==0) return 0;

  double* a = new double[m*m];
  double* b = new double[m*m];
  f(m,a,b);

  for (int i=0; i<8; ++i) {
    g(m/2);
  }
}
```

Schätzen Sie die Laufzeiten beider Funktionen mit der Θ-Notation ab.