Robert Legenstein
Helmut Hauser 28.02.2005

Prüfung zur Lehrveranstaltung Datenstrukturen und Algorithmen

WS 2004/05

Es sind keinerlei Unterlagen oder Hilfsmittel erlaubt. Es dürfen nur einzelne, lose Blätter verwendet werden! Auf jedem Blatt muss der Name und die Matrikelnummer angegeben werden! Reine Arbeitszeit beträgt **90 Minuten.**

1. [11 Punkte] Nehmen Sie an, dass die Elemente eines linearen Feldes $A[1..n]$ schon folgendermaßen teilsortiert sind:

$$A[1] \geq A[2] \quad \geq \ldots \quad \geq A[k]$$

$$A[k+1] \leq A[k+2] \quad \leq \ldots \quad \leq A[n]$$

Des weiteren sei $A[k] \geqslant A[n]$ wobei k *unbekannt* ist.

- Wie lange benötigt MERGESORT *ordnungsmäßig* zum Sortieren dieses Feldes in Abhängigkeit von n und k?
- Wie lange benötigt INSERTION-SORT *ordnungsmäßig* zum Sortieren dieses Feldes in Abhängigkeit von n und k?
- Geben Sie einen Algorithmus an (Pseudocode), der das Feld in linearer Zeit sortiert.

2. [10 Punkte] Definieren Sie das Union-Find Problem, und geben Sie eine effiziente Methode an, die Find-Operationen favorisiert. Analysieren Sie deren Zeitbedarf.

3. [8 Punkte] Geben Sie je ein einfaches *iteratives* und ein einfaches *rekursives* Programm (Pseudocode) zur Berechnung von $n!$ an. Analysieren Sie Zeit- und Speicherbedarf beider Varianten und vergleichen Sie diese.

4. [11 Punkte] Zwei Supermarktketten (A und B) haben sich geeinigt in einem noch unberührten Land ihre Läden folgendermaßen zu errichten:

- in jedem Ort soll es nur einen Supermarkt geben
- sind zwei Orte benachbart, so sollen die Supermärkte je zu einer Firma gehören

Sie sollen nun einen Algorithmus (möglichst detailierter Pseudocode + Erklärung) entwerfen, welcher überprüft, ob eine solche Aufteilung möglich ist. Zu jedem Ort x des Landes gibt es eine Liste $x \rightarrow n[1..k]$ der k Nachbarorte. Sie können beliebige Zusatzinformationen in Zusammenhang mit einem Ort speichern. Zum Beispiel könnte $x \rightarrow m$ angeben zu welcher Supermarktkette (A oder B) der Supermarkt im Ort x schlußendlich gehört. Weiters dürfen Sie die drei Funktionen *push(S,x)*, *pop(S)* und *is_empty(S)* für einen beliebigen Stack S verwenden.

Viel Erfolg!