Institut für Informatik
und Praktische Mathematik
der Universität Kiel

Kiel, 19. April 2000

Übungen zu Informatik IV
SS 2000

Serie 1

Abgabetermin: Dienstag, 02 05 00 in der Übung

Aufgabe 1 (7 Punkte)

Gegeben sei eine beliebige binäre Baumstruktur, deren Blattknoten genzzahlige Werte enthalten. Spezifizieren Sie in der in der Vorlesung eingeführten Notation einen rekursiven Algorithmus, der die kleinste dieser Zahlen bestimmt und alle Blattknoten der Struktur durch diese Zahl ersetzt (bzw. einen strukturgleichen Baum mit dieser Zahl als Blattknoten erzeugt). Wie groß ist die Komplexität des gefundenen Algorithmus, gemessen in rekursiven Funktionsaufrufen, und wieviele Durchläufe durch die Struktur erfordert er?

Hinweis: Binäre Baumstrukturen sind von der Form

```
tree = number | < tree tree >,
```

und für einen Test auf die Existenz eines atomaren Terms gibt es die Funktion `is_atom` mit der Bedeutung

```
                 { true,    if e = number
(is_atom e)  =   {
                 { false,   if e = < ... >
```

Aufgabe 1 (8 Punkte)

Spezifizieren Sie unter Benutzung der in der Vorlesung eingefuehrten Notation Algorithmen, die eine lineare Liste ganzzahliger Werte gemäß den hinreichend bekannten sog. bubble sort bzw. quicksort Verfahren in monoton aufsteigender Ordnung sortieren. Bestimmen Sie für beide Algorithmen die Zeitkomplexitäten, gemessen in rekursiven Funktionsaufrufen, für den günstigsten (best case), mittleren (average case) und ungünstigsten (worst case) Fall.

Hinweise: benutzen Sie für Operationen auf Listen die primitiven Funktionen

```
                 { a_1,         if list = < a_1 a_2 .. a_n > | < a_1 >
(head list)  =   {
                 { undefined,   otherwise

                 { < a_2 .. a_n > | <> ,   if list = < a_1 a_2 .. a_n > | < a_1 >
(tail list)  =   {
                 { undefined,              otherwise

                     { true ,       if list = < >
(is_empty list)  =   { false ,      if list = < a_1 .. >
                     { undefined,   otherwise

( < a_1 .. a_n > ++ < b_1 .. b_m > ) = < a_1 .. a_n b_1 .. b_m >
```