TECHNISCHE UNIVERSITÄT BRAUNSCHWEIG

INSTITUT FÜR

BETRIEBSSYSTEME UND RECHNERVERBUND

Prof. Dr. H. Langendörfer · Prof. Dr. M. Zitterbart

Übungen zu Algorithmen und Datenstrukturen I+II (WS9798, SS98)

1. Aufgabe: Programmverifikation

Gegeben sei der folgende Ausschnitt aus einem JAVA-Programm. Beweisen Sie die totale Korrektheit mit Hilfe des wp-Kalküls. Geben Sie zuerst die Schleifeninvariante Q an.

```
int x, y1, y2, z;
```

$\{ P \equiv (x \geq 0) \}$

```
z = 0;
y1 = 1; y2 = 1;
while (y2 <= x)
{
   z += 1;
   y1 += 2;
   y2 += y1;
}
```

$\{ z^2 \leq x \leq (z+1)^2 \}$

2. Aufgabe: Dynamisches Programmieren

Gegeben ist eine endliche Menge A von positiven natürlichen Zahlen, die aufsteigend sortiert ist. Weiterhin sei eine positive natürliche Zahl n gegeben.
Gesucht ist ein auf dem dynamischen Programmieren basierender Algorithmus, der entscheidet, ob n in Summanden aus A zerlegt werden kann. Enthält beispielsweise A die Elemente 2 und 3 ($A = \{2, 3\}$), so kann $n = 7$ als 2+2+3 gebildet werden.

Geben Sie einen Algorithmus in Pseudo-Code an, der das Programm löst. Bauen Sie hierzu schrittweise eine boolsche Matrix M auf, die an der Position $M_{i,j}$ den Wert false enthält, falls j nicht als Summanden aus den ersten i Elementen von A gebildet werden kann, und true sonst. Das Ergebnis soll in $M_{l,n}$ stehen ($l = |A|$).

3. Aufgabe: Baumstrukturen

In einen leeren Binärbaum werden Knoten mit den Schlüsseln $\{308, 52, 600, 987, 51, 4, 222, 412, 999, 577, 578\}$ eingefügt.

(a) Während des Aufbaus geht die (anfangs vorhandene) AVL-Eigenschaft des Baumes verloren. Geben Sie den Baum zu dem Zeitpunkt an, wo sie gerade wiedergewonnen ist.

(b) Auch die Eigenschaft des Baumes, vollständig ausgeglichen zu sein, geht während des Aufbaus verloren. Wird der Baum danach jemals wieder vollständig ausgeglichen? Wenn ja, nach welcher Eingabe? Wenn nein, durch welchen Schlüssel hat er die Eigenschaft verloren?

4. Aufgabe: Sortierverfahren

Gegeben sei ein Array, welches die Zahlen $\{17, 24, 3, 1, 12, 7, 9, 5, 2\}$ enthält.

(a) Sortieren Sie dieses Array mittels Quicksort und geben Sie die Arrayausprägung nach jedem Sortierschritt an.

(b) Ein Sortierverfahren wird als *stabil* bezeichnet, wenn sich die relative Reihenfolge gleicher Schlüssel beim Sortieren nicht ändert. Stellt Quicksort ein stabiles Sortierverfahren dar (Begründung) ?

*Die **Vordiplomklausur** findet am Dienstag, 16. März 1999, von 13^{30} – 17^{30} im AudiMax und PK 15.1 statt.*

*Als Hilfsmittel sind ausschließlich Unterlagen der Vorlesung zugelassen, d.h. eigene Mitschriften sowie Kopien der Folien. **Nicht** zugelassen sind weitere Literatur (Bücher etc) sowie jedwede Form von elektronischen Hilfsmitteln.*

Lösungen

Lösung zur 1. Aufgabe

$P \equiv (x \geq 0)$
$N \equiv (z^2 \leq x \leq (z+1)^2)$
$Q \equiv (z^2 \leq x \wedge y_1 = 2z+1 \wedge y_2 = (z+1)^2)$

(1) z.z.: $P \Rightarrow wP(„z=0, y_1=1, y_2=1“, Q)$
$wP(„z=0, y_1=1, y_2=1“, z^2 \leq x \wedge y_1 = 2z+1 \wedge y_2 = (z+1)^2)$
$\equiv wP(„z=0, y_1=1“, z^2 \leq x \wedge y_1 = 2z+1 \wedge 1 = (z+1)^2)$
$\equiv wP(„z=0“, z^2 \leq x \wedge 1 = 2z+1 \wedge 1 = (z+1)^2)$
$\equiv 0^2 \leq x \wedge 1 = 1 \wedge 1 = 1^2$
$\equiv x \geq 0 \wedge TRUE$ folgt aus P $\square$

(2) z.z.: $Q \wedge (y_2 \leq x) \Rightarrow wP(„z+=1, y_1+=2, y_2+=y_1“, Q)$
$wP(„z+=1, y_1+=2, y_2+=y_1“, z^2 \leq x \wedge y_1 = 2z+1 \wedge y_2 = (z+1)^2)$
$\equiv wP(„z+=1, y_1+=2“, z^2 \leq x \wedge y_1 = 2z+1 \wedge y_2 + y_1 = (z+1)^2)$
$\equiv wP(„z+=1“, z^2 \leq x \wedge y_1 + 2 = 2z+1 \wedge y_2 + y_1 + 2 = (z+1)^2)$
$\equiv (z+1)^2 \leq x \wedge y_1 + 2 = 2(z+1)+1 \wedge y_2 + y_1 + 2 = (z+2)^2$
$\Leftrightarrow (z+1)^2 \leq x \wedge y_1 = 2z+1 \wedge y_2 + 2z + 1 + 2 = z^2 + 4z + 4$
$\Leftrightarrow (z+1)^2 \leq x \wedge y_1 = 2z+1 \wedge y_2 = (z+1)^2$
folgt aus $Q \wedge (y_2 \leq x)$ $\square$

(3) z.z.: $Q \wedge (y_2 > x) \Rightarrow N$
$(z^2 \leq x \wedge y_1 = 2z+1 \wedge y_2 = (z+1)^2 \wedge y_2 > x) \Rightarrow z^2 \leq x \leq (z+1)^2$
klar $\square$

(4) Terminiertheit: y2 wird mit 1 initialisiert und pro Schleifendurchlauf um einen positiven Wert erhöht; da x in der Schleife seinen Wert nie ändert (mithin konstant ist), ist y2 irgendwann größer als x, entsprechend terminiert die Schleife.

Lösung zur 2. Aufgabe

Eine *mögliche* Lösung:

Konstruiere $l \times n$-Matrix M mit

$$M_{i,j} := \begin{cases} false & ; \quad \text{falls j nicht als Summanden der i ersten Elem. von A gebildet werden kann} \\ true & ; \quad sonst \end{cases}$$

Initialisiere $M_{i,j}$ mit $false$

```
for i = 1 to l do
  for j = 1 to n do
    if ( ∃a_k(1 ≤ k ≤ i) : a_k = j ) then M_{i,j} := true
    else if M_{i-1,j} (für i>1) then M_{i,j} := true
    else
      r=1
      repeat
        M_{i,j} := (M_{i,r} and M_{i,j-r})
        r=r+1
      until M_{i,j} or (r > i/2)
if (M_{l,n}) then „ja“ else „nein“
```

Lösung zur 3. Aufgabe

(a) Bis zum Einfügen des Schlüssels 4 besitzt der Baum die *AVL*-Eigenschaft, nach Einfügen von 222 ist diese wiederhergestellt und geht durch das Einfügen von 578 wieder verloren.

Lösung zur 4. Aufgabe

(a) %

(b) Quicksort stellt *kein* stabiles Verfahren dar.
Gegenbeispiel: $\{3, 3, 1\}$ (die Wahl des Pivotelements spielt keine Rolle)