

Norman May

B6, 29, Raum C0.05

68131 Mannheim

Telefon: (0621) 181-2517

Email: norman@pi3.informatik.uni-mannheim.de

Matthias Brantner

B6, 29, Raum C0.05

68131 Mannheim

Telefon: (0621) 181-2517

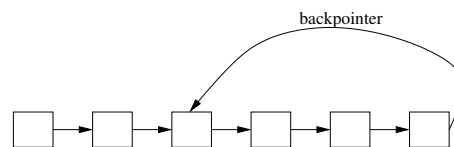
Email: msb@pi3.informatik.uni-mannheim.de

Algorithmen und Datenstrukturen
Wintersemester 2004/057. Lösungsvorschlag
17. Dezember 2004

Aufgabe 1

5 Punkte

Gegeben eine einfach verkettete Liste. Der `next`-Zeiger des letzten Elements der Liste ist entweder `nil` oder ein sog. Backpointer. Zur Veranschaulichung dient folgende Grafik.



Geben sie einen Algorithmus mit linearer Laufzeit in der Länge der Liste und konstantem Speicherbedarf an, der entscheidet, ob die Liste einen Backpointer enthält. Die Liste darf dabei nicht verändert oder zerstört werden.

Lösung

```
BACKPOINTER (l)
1 iter1 ← list.head
2 iter2 ← list.head
3 while (iter2 ≠ nil)
4   iter2 ← iter2.next
5   if (iter2 = iter1)
6     return true
7   if (iter2 ≠ nil)
8     iter2 ← iter2.next
9     if (iter2 = iter1)
10      return true
11   iter1 ← iter1.next
12 else
13   return false
14 return false
```

Der Algorithmus benützt zwei Zeiger, um durch die Liste zu gehen. Der erste Zeiger geht in jedem Schritt zwei Elemente weiter. Der zweite Zeiger im gleichen Schritt nur einen Schritt. Sobald der zweite Zeiger auf den ersten Trifft enthält die Liste einen backpointer.

Aufgabe 2

9 Punkte

Es sollen die Schlüssel 10, 22, 31, 4, 15, 28, 17, 88, 59 mittels offener Adressierung in eine Hash-Tabelle eingetragen werden mit $m = 11$. Die primäre Hash-Funktion lautet $h(k) = k \bmod m$. Geben Sie jeweils die Hash-Tabelle nach dem Einfügen der Schlüssel für folgende Verfahren zur Konfliktauflösung an.

Aufgabe 2 a)

3 Punkte

Lineares Hashing

Aufgabe 2 b)

3 Punkte

Quadratisches Hashing mit $c_1 = 1$ und $c_2 = 3$

Aufgabe 2 c)

3 Punkte

Doppeltes Hashing mit $h_2(k) = 1 + (k \bmod (m - 1))$

Lösung

Lineares Hashing (2 Punkte für die einzelnen Schritte)

k: 10 i: 0 j: 10
 k: 22 i: 0 j: 0
 k: 31 i: 0 j: 9
 k: 4 i: 0 j: 4
 k: 15 i: 0 j: 4
 k: 15 i: 1 j: 5
 k: 28 i: 0 j: 6
 k: 17 i: 0 j: 6
 k: 17 i: 1 j: 7
 k: 88 i: 0 j: 0
 k: 88 i: 1 j: 1
 k: 59 i: 0 j: 4
 k: 59 i: 1 j: 5
 k: 59 i: 2 j: 6
 k: 59 i: 3 j: 7
 k: 59 i: 4 j: 8

Hashtabelle: 22 88 0 0 4 15 28 17 59 31 10 (1 Punkt)

Quadratische Hashing (2 Punkte für die einzelnen Schritte)

k: 10 i: 0 j: 10
k: 22 i: 0 j: 0
k: 31 i: 0 j: 9
k: 4 i: 0 j: 4
k: 15 i: 0 j: 4
k: 15 i: 1 j: 8
k: 28 i: 0 j: 6
k: 17 i: 0 j: 6
k: 17 i: 1 j: 10
k: 17 i: 2 j: 9
k: 17 i: 3 j: 3
k: 88 i: 0 j: 0
k: 88 i: 1 j: 4
k: 88 i: 2 j: 3
k: 88 i: 3 j: 8
k: 88 i: 4 j: 8
k: 88 i: 5 j: 3
k: 88 i: 6 j: 4
k: 88 i: 7 j: 0
k: 88 i: 8 j: 2
k: 59 i: 0 j: 4
k: 59 i: 1 j: 8
k: 59 i: 2 j: 7
Hashtabelle: 22 0 88 17 4 0 28 59 15 31 10 (1 Punkt)

Doppeltes Hashing (2 Punkte für die einzelnen Schritte)

k: 10 i: 0 j: 10
k: 22 i: 0 j: 0
k: 31 i: 0 j: 9
k: 4 i: 0 j: 4
k: 15 i: 0 j: 4
k: 15 i: 1 j: 10
k: 15 i: 2 j: 5
k: 28 i: 0 j: 6
k: 17 i: 0 j: 6
k: 17 i: 1 j: 3
k: 88 i: 0 j: 0
k: 88 i: 1 j: 9
k: 88 i: 2 j: 7
k: 59 i: 0 j: 4
k: 59 i: 1 j: 3
k: 59 i: 2 j: 2
Hashtabelle: 22 0 59 17 4 15 28 88 0 31 10 (1 Punkt)

Bei der Verwendung von doppeltem Hashing zur Konfliktauflösung lautet die Hash-Funktion

$$h(k, i) = (h_1(k) + ih_2(k)) \bmod m$$

. Zeigen Sie, daß die Probensequenz

$$\langle h(k, 0), h(k, 1), \dots, h(k, m-1) \rangle$$

genau dann eine Permutation der Slot-Sequenz $\langle 0, 1, \dots, m-1 \rangle$ ist, wenn $h_2(k)$ und m teilerfremd sind.

Lösung

Zu zeigen $\langle h(k, 0), \dots, h(k, m-1) \rangle$ eine Permutation von $\langle 0, \dots, m-1 \rangle$ gdw. $ggT(h_2(k), m) = 1$.

\Rightarrow

Widerspruchsbeweis (2.5 Punkte)

Sei $\langle h(k, 0), \dots, h(k, m-1) \rangle$ eine Permutation und $ggT(h_2(k), m) = d, d > 1$.

Sei $m = yd$ und $h_2(k) = xd$. (0.5 Punkte)

$$\begin{aligned} h(k, i) &= (h_1(k) + ih_2(k)) \bmod m \\ &= h_1(k) + ih_2(k) - \left\lfloor \frac{h_1(k) + ih_2(k)}{m} \right\rfloor m \\ &= h_1(k) + ixd - \left\lfloor \frac{h_1(k) + ih_2(k)}{m} \right\rfloor yd \\ &= h_1(k) + cd, \text{ mit } c \in \mathbb{Z} \end{aligned}$$

Da c ganzzahlig, kann mit $h_1(k) + cd$ keine Permutation gebildet werden. (2 Punkte)

\Leftarrow

Widerspruchsbeweis (2.5 Punkte)

Sei $ggT(h_2(k), m) = 1$ und $\langle h(k, 0), \dots, h(k, m-1) \rangle$ keine Permutation. (0.5 Punkte)

Da $\langle h(k, 0), \dots, h(k, m-1) \rangle$ keine Permutation und $0 \leq h(k, i) \leq m-1$, existiert mindestens ein Paar $i_1, i_2 \in 0, \dots, m-1$ mit $i_1 \neq i_2$ und $h(k, i_1) = h(k, i_2)$. Damit ist $h(k, i_1) - h(k, i_2)$ teilbar durch m .

$$\begin{aligned} h(k, i_1) - h(k, i_2) \bmod m &= 0 \\ \Leftrightarrow (i_1 - i_2)h_2(k) \bmod m &= 0 \\ \Leftrightarrow (i_1 - i_2)h_2(k) = xm \text{ mit } x \in \mathbb{Z} \end{aligned}$$

Da, $(i_1 - i_2) < m$, ist das $kgv(h_2(k), m) < h_2(k)m$. Dies geht aber nur, wenn der $ggT(h_2(k), m) > 1$ ist. (2 Punkte)

Ein Inorder-Durchlauf für einen binären Suchbaum mit n Knoten kann implementiert werden, indem man zunächst das Minimum mittels der Funktion *Tree-Minimum* bestimmt und anschließend $n-1$ mal *Tree-Successor* auf dem aktuellen Knoten aufgerufen wird. Zeigen Sie, dass die Laufzeit des Algorithmus in $\Theta(n)$ liegt.

Lösung

MS-Tree-Traversal(T) (2 Punkte)

```

 $x \leftarrow \text{Tree-Minimum}(T)$ 
for  $i \leftarrow 1$  to  $n - 1$ 
  do  $x \leftarrow \text{Tree-Successor}(x)$ 

```

Anzahl der Zugriffe auf einen Knoten x ermitteln: (2 Punkte)

- Tree-Minimum
- Tree-Successor(left[x])
- Tree-Successor(x)
- Tree-Successor(right[x])

Somit ergibt sich eine konstante Anzahl von Zugriffen pro Knoten. Sei n die Anzahl der Knoten. Die Laufzeit beträgt somit $O(n)$. (1 Punkt)

Sei T ein binärer Suchbaum, x ein Blatt in T und y der Vaterknoten von x . Zeigen Sie, dass $key[y]$ entweder der kleinste Schlüssel größer $key[x]$ oder der größte Schlüssel kleiner $key[x]$ ist.

Lösung

1. Zu zeigen, sei x ein Blatt und $y = p[x]$, dann gilt: $x < y \Rightarrow \neg \exists z : x < z < y$
Widerspruchsbeweis (2 Punkte)

Annahme: $x < y \wedge \exists z : x < z < y$

Es folgt, dass z entsprechend der In-Order-Reihenfolge vor y im Baum und nach x enthalten sein muss. Dies geht aber nur, wenn z sich im rechten Teilbaum unterhalb von x befindet, der aber leer ist.

2. Zu zeigen, sei x ein Blatt und $y = p[x]$, dann gilt: $x > y \Rightarrow \neg \exists z : y < z < x$
Widerspruchsbeweis (2 Punkte)

Annahme $x > y \wedge \exists z : y < z < x$

Es folgt, dass z entsprechend der In-Order-Reihenfolge vor x und nach y im Baum enthalten sein muss. Dies ist aber nur möglich, falls z sich im linken Teilbaum unterhalb von x befindet, der aber leer ist.