Technische Universität München
Institut für Informatik
Lehrstuhl für Effiziente Algorithmen
Prof. Dr. Ernst W. Mayr
Dr. Jens Ernst

WS 2003/2004
Mittelklausur
19. Dezember 2003

Grundlegende Algorithmen

Aufgabe 1 (8 Punkte)

Beweisen oder widerlegen Sie folgende Aussagen, wobei "ln" den natürlichen Logarithmus bezeichnet:

(a) $\sqrt{\ln n} = o(\ln(\sqrt{n}))$

(b) $\ln(n \ln n) = \omega(\log_2 n + \log_2(\log_2 n))$

(c) $(\ln n)^{\ln n}$ ist superpolynomiell in n

(d) $n^{\ln n}$ ist subexponentiell in n

Lösungsvorschlag:

(a) Wahr

$$\lim_{n\to\infty} \frac{\sqrt{\ln n}}{\ln \sqrt{n}} = \lim_{n\to\infty} \frac{\frac{1}{2\sqrt{\ln n}}\frac{1}{n}}{\frac{1}{\sqrt{n}}\frac{1}{2\sqrt{n}}} = \lim_{n\to\infty} \frac{1}{\sqrt{\ln n}} = 0$$

oder: $\ln \sqrt{n} = \frac{1}{2}\ln n$, und $\sqrt{\ln n} = o(\frac{1}{2}\ln n)$

(b) Falsch

$$\begin{aligned} \ln(n\ln(n)) &= \ln n + \ln\ln n = \frac{\log_2 n}{\log_2 e} + \frac{1}{\log_2 e}\log_2\left(\frac{\log_2 n}{\log_2 e}\right) \\ &= \Theta(\log_2 n) + \Theta(\log_2\log_2 n - \log_2\log_2 e) \neq \omega(\log_2 n + \log_2(\log_2 n)) \end{aligned}$$

(c) Wahr

$$(\ln n)^{\ln n} = e^{\ln((\ln n)^{(\ln n)})} = e^{\ln n \ln\ln n} = \left(e^{\ln n}\right)^{\ln\ln n} = n^{\ln\ln n} = \omega(n^k) \text{ für jedes } k \geq 0$$

(d) Wahr

$$n^{\ln n} = 2^{\log_2(n^{\ln n})} = 2^{\ln n \log_2 n} = 2^{o(\varepsilon n)} = o(2^{\varepsilon n}) \text{ für jedes } \varepsilon > 0.$$

Aufgabe 2 (8 Punkte)

Geben Sie ein RAM-Programm an, das die Funktion $f : \mathbb{N} \to \mathbb{N}_0$ mit $f(n) := \lfloor \log_2 n \rfloor$ berechnet. Hierbei gilt die übliche Konvention, daß zu Beginn der Ausführung das Argument n in Register `c(1)` steht (wobei $n > 0$ garantiert ist), während alle übrigen Register den Wert 0 haben. Das Resultat $f(n)$ soll am Ende ebenfalls in Register `c(1)` stehen.
Begründen Sie kurz Ihr Vorgehen, wobei Sie Resultate aus der Vorlesung verwenden dürfen.

Lösungsvorschlag:

Man halbiere den Ausgangswert n solange durch SHIFT-Operationen, bis der Wert 0 erreicht ist. Die Anzahl der Halbierungen ist $\lfloor \log_2 n \rfloor + 1$. RAM-Programm:

```
 1: LOAD #0    // initialisiere c(2) als Zaehler
 2: STORE 2
 3: LOAD 1     // Schleifenanfang
 4: IF =#0 THEN GOTO 11 // Ende erreicht
 5: SHIFT #1    // halbiere c(1) mit Abrunden
 6: STORE 1
 7: LOAD 2     // erhoehe Zaehler
 8: ADD #1
 9: STORE 2
10: GOTO 3     // Schleifenende
11: LOAD 2     // Dekrementiere Zaehler
12: SUB #1
13: STORE 1    // Resultat
14: END
```

Aufgabe 3 (8 Punkte)

Geben Sie einen Algorithmus an, der die in einem binären Suchbaum gespeicherten Schlüssel in aufsteigend sortierter Reihenfolge ausgibt. Die Laufzeit soll linear in der Anzahl der Knoten des Baums sein. Der Baum sei durch einen Pointer auf die Wurzel gegeben, und an jedem Knoten seien Pointer auf dessen Kind(er) gespeichert.

Lösungsvorschlag:

Sei `t` der Pointer auf den Baum, und jeder Knoten `v` habe die beiden Pointer `v.left`, `v.right` auf eventuelle Kinder, sowie den Schlüssel `v.key`. Induktiver Ansatz:

- **Base Case:** Falls der Baum leer ist, ist nichts zu tun. Falls der Baum nur einen Knoten `*t` besitzt, gib `t->key` aus.
- **Induktionsschritt:** Da alle Knoten im linken Unterbaum kleinere Schlüssel haben als die Wurzel `*t` und alle Knoten im rechten Unterbaum größere Schlüssel haben als die Wurzel, kann man induktiv wie folgt vorgehen: Man gebe die Schlüssel in `t->left` sortiert aus (per Induktion), dann `t->key`, dann die Schlüssel in `t->right` ebenfalls sortiert (per Induktion).

Der Algorithmus lautet also wie folgt:

```
void output(tree* t){
  if (t=NULL) then return;
  elsif (t->left=NULL and t->right=NULL) then
       print t->key;
       return;
  fi;
  output(t->left);
  output(t->right);
}
```

Aufgabe 4 (10 Punkte)

Für Heaps, in denen ganzzahlige Schlüsselwerte gespeichert sind, soll eine neue Operation `increase_min` eingeführt werden, die den minimalen Schlüsselwert eines Heaps h um den nichtnegativen Wert d erhöht und sicherstellt, daß die Heapbedingung am Ende wieder erfüllt ist. (Die Heapbedingung verlangt für jeden Knoten v des Heaps, daß die Schlüssel der Kinder von v nicht kleiner sind als der Schlüssel von v.) Es sei h als Baum implementiert und durch einen Pointer auf die Wurzel gegeben. An jedem Knoten seien Pointer auf dessen Kind(er) gespeichert.

(a) Geben Sie einen möglichst effizienten Algorithmus an, der diese Operation realisiert. Die Heap-Operationen aus der Vorlesung (d.h. `create_heap`, `delete_min` und `reheap`) dürfen nicht explizit verwendet werden.

(b) Geben Sie eine möglichst gute obere Schranke (in Landau-Notation) für die Anzahl der Schritte in Abhängigkeit von der Anzahl der Elemente im Heap an. Begründen Sie Ihre Aussage.

Lösungsvorschlag:

(a) Zunächst vergrößert man den Schlüssel an der Wurzel, wodurch die Heapbedingung i.a. verletzt wird. Danach wird, beginnend bei der Wurzel, am jeweils aktuellen Knoten der Schlüssel mit dem kleinsten Schlüssel der Kinderknoten vertauscht, bis die Heapbedingung wieder erfüllt ist. Letzteres ist eine `reheap`-Operation.

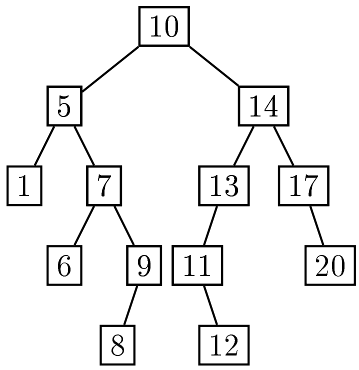
```
void increase_min(heap *h, unsigned d){
  h->key-=d;
  v:=h;
  while ((v->left!=NULL && v->key>v->left->key) ||
         (v->right!=NULL && v->key>v->right->key)) do
    if    (v->left!=NULL && v->right!=NULL && v->left<=v->right) then w:=v->left;
    elsif (v->left!=NULL && v->right!=NULL && v->left>v->right) then w:=v->right;
    elsif (v->left!=NULL) then w:=v->left;
    else w:=v->right;
```

```
      fi;
      vertausche v->key mit w->key;
      v:=w;
    od;
  }
```

(b) Pro besuchtem Level im Baum werden $O(1)$ Schritte ausgeführt. Die Anzahl der besuchten Levels ist beschränkt durch $O(\log_2 n)$, da ein Heap ein fast vollständiger Binärbaum ist. Somit sind die Gesamtkosten beschränkt durch $O(\log_2 n)$.

Aufgabe 5 (6 Punkte)

Gegeben sei der folgende binäre Suchbaum t.



Beschreiben Sie kurz, was bei der Ausführung der Operation `delete(10)` geschieht, und geben Sie an, wie t nach Ausführung dieser Operation aussieht.

Lösungsvorschlag:

Der Knoten mit Schlüsselwert 10 wird durch `is_element` gesucht und an der Wurzel gefunden. Dann wird in den rechten Unterbaum gegangen und dort so lange nach links verzweigt, bis der aktuelle Knoten kein linkes Kind mehr hat. Dies ist der Knoten w mit Schlüssel 11. Dieser Schlüssel ersetzt den Schlüssel der Wurzel. Schliesslich muss der Knoten w gelöscht werden, was dadurch geschieht, dass er durch sein einziges Kind ersetzt wird. Es entsteht der Baum