Universität des Saarlandes
FR 6.2 Informatik
Prof. Dr. Hans-Peter Lenhof
Dipl. Inform. Andreas Hildebrandt

11. Übung zur Vorlesung Programmierung II, SS 2003

Hinweis: Dies ist das letzte Übungsblatt zur Vorlesung Programmierung II in diesem Semester. Gleichzeitig handelt es sich um eine Probeklausur, mit der Sie sich auf die Klausur am 19.07. vorbereiten können. Die Aufgaben die mit 0 Punkten versehen sind gehen nicht in die Wertung ein, werden aber trotzdem korrigiert. *Geben Sie diesmal jede Aufgabe auf einem eigenen Blatt ab, und versehen Sie jedes Blatt mit Ihrem Namen, Ihrer Matrikelnummer und dem Namen Ihres Übungsgruppenleiters! Geben Sie bei jeder Aufgabe bitte auch mit an, wieviel Zeit Sie zur Bearbeitung benötigt haben.*
Da Sie dieses Blatt unter Klausurbedingungen bearbeiten sollten, sollten Sie die C++-Implementierungsaufgaben diesmal nicht am Computer sondern nur auf dem Papier lösen.

Aufgabe 1: Wissenstest (0 Punkte)

(a) Erklären Sie *kurz* anhand eines `vector` das Konzept einer `C++` Template-Klasse.

(b) Definieren Sie $\Theta(n)$

(c) Geben Sie die Worst-Case Laufzeiten der Operationen *vorne anfügen* und *Suche* für eine *sortierte doppelt verkette Liste* und für ein *unsortiertes Array* an.

(d) Beschreiben Sie kurz drei Techniken zur Kollisionsauflösung bei Hashing mit *open adressing*.

(e) Definieren Sie den Begriff *Universelle Klasse von Hashfunktionen*.

(f) Was versteht man unter einem *AVL-Baum*?

(g) Was versteht man unter einem *Euler-Circuit*?

(h) Beschreiben Sie den Algorithmus zur Berechnen der starken Zusammenhangskomponenten, den Sie in der Vorlesung kennengelernt haben (Sie brauchen hier keinen Code anzugeben!).

Zusatzaufgabe 2: C++ (10 Bonuspunkte)
Erstellen Sie eine *C++*-Templateklasse `Baum24<T>` die einen 2–4 - Baum implementiert. Geben Sie dazu die Klassendeklaration möglichst vollständig an (inklusive Konstruktor, Copykonstruktor, Destruktor und Assignmentoperator sowie allen Operationen die Sie bei einem 2–4 - Baum erwarten würden). Geben Sie zusätzlich die Implementierung der

Methode `bool suche(const T& i)` an, welche zurückliefert, ob das Element i im Baum enthalten ist.

Zusatzaufgabe 3: $\mathcal{O}$-Notation (10 Bonuspunkte)

- Sei $f(n) = \sum_{i=0}^{k} g_i(n)$, und es gelte für einen Index $i \in \{0, \ldots, k\} : \forall j \neq i, j \in \{0, \ldots, k\} : g_j(n) \in \mathcal{O}(g_i(n))$. Zeigen Sie: $f(n) \in \Theta(g_i(n))$,
- Finden Sie eine Anordnung der folgenden Funktionen, so dass gilt $\forall i : f_i \in \mathcal{O}(f_{i+1})$:
  (a) n
  (b) $(\sqrt{2})^{\log_2(n)}$
  (c) $(\frac{3}{2})^n$
  (d) $\log_2 \log_2(n)$
  (e) $(\log_2(n))^2$

Aufgabe 4: Rekurrenzgleichungen, Substitutionsmethode (10 Punkte)
Lösen Sie die folgende Rekurrenzgleichung mit Hilfe der Substitutionsmethode (zeigen Sie dabei nicht nur "$\mathcal{O}$" sondern "Θ"). Zeigen Sie die Korrektheit Ihrer Lösung durch vollständige Induktion.

$$T(n) = 3T(\frac{n}{5}) + cn^2$$

Zusatzaufgabe 5: Mastertheorem (10 Bonuspunkte)
Zeigen Sie für jede der folgenden Rekurrenzen ob das Mastertheorem zur Lösung angewandt werden kann. Bestimmen Sie in den Fällen, in denen sich das Theorem anwenden lässt, die Lösung der Rekurrenz.

- $T(n) = 49T(\frac{n}{7}) + cn^2 + d$
- $T(n) = 63T(\frac{n}{4}) + 17n^3$
- $T(n) = 64T(\frac{n}{8}) + n\ln(n) + n$

Zusatzaufgabe 6: Heaps (10 Bonuspunkte)
Gegeben seien zwei binäre Heaps A und B, die nicht durch Arrays, sondern durch Listen repräsentiert werden (d.h. jeder Knoten hat Zeiger auf seine beiden Kinder). A bestehe aus n Elementen, B aus m. Entwickeln Sie einen Algorithmus der A und B in $\mathcal{O}(\log(m+n))$ (worst-case) zu einem einzigen Heap vereinigt, der alle Elemente von A und B enthält. Verwenden Sie dabei C++–Syntax.

Aufgabe 7: Eulergraphen (10 Punkte)
Gegeben sei ein ungerichteter Eulergraph $G = (V, E)$. Geben Sie in C++–Syntax einen Algorithmus an, der einen Kreis in G findet. Begründen Sie die Korrektheit Ihres Algorithmus. Sie dürfen in dieser Aufgabe auch auf `LEDA` – Datenstrukturen zurückgreifen.

Aufgabe 8: Erwartungswerte (10 Punkte)
Berechnen Sie den Erwartungswert und die Varianz einer *geometrisch verteilten* Zufallsvariablen, d.h. einer Variablen X mit $p(\{X = i\}) = (1-p)^{i-1}p$.

Aufgabe 9: Suche (10 Punkte)
Es sei $A = \{a_0, \ldots, a_{n-1}\}$ ein `vector<int>` der Länge n. A habe zusätzlich die Eigenschaft, dass $|A[i] - A[i+1]| \leq 1 \forall i \in \{0, \ldots, n-2\}$. Ausserdem gelte $x := A[0] < A[n-1] =: y$. Entwickeln Sie einen möglichst effizienten Algorithmus, der für eine Integerzahl z mit $x \leq z \leq y$ einen Index $j \in \{0, \ldots, n-1\}$ zurückliefert, für den $A[j] = z$ gilt.
Wie viele Vergleiche mit z führt Ihr Algorithmus im worst-case durch?

Abgabe: 11.07.2003