Universität Stuttgart
Fakultät 5, Institut FMI
Einführung in die Informatik II

Sommersemester 2006
Claus, Draskoczy, Schmid
30.06.06

Aufgabenblatt 10 **Abgabe: 08.07.06, 24.00 Uhr**

---

1. Sortierverfahren I (leicht) (schriftlich, 1+1 Punkte)
Sortieren Sie die Zahlenfolge 12, 17, 1, 19, 23, 8, 3, 42 und 2 mit den Verfahren *Fachverteilen* und *Minimumsuche*. Geben Sie im eClaus-System die Zahlenfolge für jeden Zwischenschritt ab!

2. Heapsort (mittel) (schriftlich, 1+1+1+1+1 Punkte)
Das Sortierverfahren Heapsort basiert auf "absteigenden Heaps", d.h., das größte Element steht in der Wurzel.

a) Heaps sind spezielle Bäume. Bei der Implementierung von Heaps benutzt man im Verfahren "Heapsort" jedoch statt einer Zeigerstruktur ein Array für die Elemente. Begründen Sie dieses Vorgehen.

b) Wie viele Elemente muss das Array mindestens umfassen, um darin einen Heap mit n Elementen abspeichern zu können?
☐ n Elemente ☐ $2n$ Elemente ☐ $n \cdot \log n$ Elemente ☐ 2^n Elemente

c) Geben Sie eine formale Beschreibung für die Heapbedingung für einen absteigenden Heap an, der in einem Array mit unterer Indexgrenze 1 gespeichert ist.

d) Die Prozedur `sink`(k,n) stellt die Heapbedingung für einen (absteigenden) Heap mit n Elementen für den Unterbaum an der Stelle k wieder her.
Gegeben ist ein Array mit den Elementen 12, 7, 23, 17, 42, 3, 5. Geben Sie an, welche Aufrufe von `sink(.,.)` Sie verwenden, um dieses Array in einen Heap zu verwandeln. Geben Sie die Aufrufe und das am Ende resultierende Array an.

e) Mit welchen Parametern wird die Prozedur `sink(.,.)` das erste Mal in der Sortierphase aufgerufen? Führen Sie diesen Schritt durch und geben Sie das resultierende Array an.

3. Quicksort (mittel) (schriftlich, 2 Punkte)
Gegeben ist ein Array mit den Elementen 12, 7, 23, 17, 42, 3, 5, 8, 22, 24. Führen Sie den Quicksort Algorithmus durch. Als Pivotelement soll dabei jeweils das links stehende Element gewählt werden. Geben Sie die rekursiven Aufrufe in der Reihenfolge, wie sie bei der Berechnung auftreten, und die Inhalte der entsprechenden Teilfelder an.

4. (Bottom-Up-)Heapsort (mittel-schwer) (schriftlich, 1+3 Punkte)
Bei Bottom-Up-Heapsort wird im Gegensatz zum normalen Heapsortverfahren zunächst der Einsinkpfad bis zum Blatt berechnet und vom Blatt aus entlang des Einsinkpfads die Stelle bestimmt, an die das Element hingehört.

a) Begründen Sie kurz, warum zu erwarten ist, dass Bottom-Up-Heapsort im Mittel weniger Vergleiche benötigt als das normale Heapsort.

b) Sind nicht alle Schlüssel verschieden (z.B. wenn Paare (Vorname,Nachname) bzgl. des Nachnamens sortiert werden), so erhält man u.U. verschiedene Reihenfolgen der Elemente in der mittels Bottom-Up-Heapsort sortierten Folge im Vergleich zu der mit dem normalen Heapsort bestimmten Folge. Konstruieren Sie ein Beispiel, das dieses Verhalten demonstriert.

5. Suche nach dem k-kleinstem Element (mittel-schwer) (schriftlich, 4 Punkte)
Um das k-kleinste Element einer Folge zu bestimmen, kann man so vorgehen, dass man das Feld sortiert und dann das Element mit Index k ausgibt. Um das k-kleinste Element zu bestimmen, ist es jedoch nicht nötig die komplette Folge zu sortieren. Modifizieren Sie den Quicksortalgorithmus so, dass er zu einem Feld $A[1..n]$ und einer Zahl k, $1 \leq k \leq n$ das k-kleinste Element bestimmt. Das Feld A kann danach in beliebiger Reihenfolge vorliegen.
Schreiben Sie Ihr Programm in Pseudocode und geben Sie zusätzlich eine ausführliche Beschreibung im eClaus-System ab.

6. Sortierverfahren II (mittel) (schriftlich, 4 Punkte)
Überlegen sie sich selbst ein Sortierverfahren (z.B. aus der Kombination verschiedener Standardverfahren)!
Implementieren Sie ihren Algorithmus in Ada und geben Sie zusätzlich eine ausführliche Beschreibung im eClaus-System ab. Die zu sortierenden Schlüssel sollen in einem Integer-Array vorliegen und zum Schluss auch in diesem Array in sortierter Form vorliegen.
Die schnellsten Algorithmen werden auf der Vorlesungsseite veröffentlicht!