Technische Universität München
Institut für Informatik
Prof. Dr. Angelika Steger

SS 2000
Übungsblatt 8
23. Juni 2000

Einführung in die Informatik IV

Aufgabensteller: Alex Hall
Abgabetermin: Mittwoch, 5. Juli 2000, 9:25 Uhr

Hinweis: Es wird insgesamt 10 Übungsblätter mit abgabepflichtigen Aufgaben geben. D.h. um zur Prüfung zugelassen zu werden, benötigt man mindestens 40 Punkte bei den Hausaufgaben.

Tutoraufgabe 1

Untersuchen Sie die worst- und average-case Laufzeit des Quickselect Algorithmus.
Quickselect funktioniert ähnlich wie Quicksort. Eingabe ist ein unsortiertes Feld mit n Elementen und eine Zahl $k \leq n$. Der Algorithmus liefert das k-te Element des sortierten Feldes (das Element mit *Rang* k). Für $k = \lfloor n/2 \rfloor$ liefert Quickselect also z.B. den Median.
Der eigentliche Algorithmus funktioniert folgendermaßen: Es wird zuerst ein beliebiges Element der Folge als Pivot ausgewählt und das Feld wie im Aufteilungsschritt von Quicksort aufgeteilt. Dabei wird der Rang r des Pivots bestimmt und nun rekursiv entweder in der linken oder in der rechten Hälfte weitergesucht, je nachdem, ob $k < r$ oder $k > r$ ist. Sollte $k = r$ sein, dann wurde natürlich das Element mit Rang k gefunden.
Für die Analyse kann davon ausgegangen werden, daß die Elemente eindeutig sind.

Tutoraufgabe 2

Es wird eine unsortierte Liste mit n eindeutigen Elementen betrachtet.

a) **Untere Schranke für die Maximumssuche:** Zeigen Sie, daß jeder Algorithmus zum Finden des Maximums in einer Liste mit n Elementen mindestens $n - 1$ Vergleiche benötigt.

b) **Untere Schranke für das Sortieren:** Ein *vergleichsbasierter Sortieralgorithmus* verwendet für seine Entscheidungen nur die Ergebnisse von Vergleichen auf den zu sortierenden Schlüsseln. Beweisen Sie, daß jeder solche Algorithmus $\Omega(n \cdot \log n)$ Vergleichsoperationen bei Eingabelänge n benötigt.

Hausaufgabe 1 (freiwillig)

Zeigen Sie, daß jeder Algorithmus zum Mischen zweier sortierter Folgen der Länge n bzw. m im worst-case mindestens

$$\left\lceil \log\left(\binom{n+m}{n}\right)\right\rceil$$

Vergleiche benötigt, wenn alle Elemente paarweise verschieden sind.
Hinweis: Verwenden Sie ein Gegenspielerargument.

Hausaufgabe 2 (freiwillig)

Betrachten Sie den Ansatz zum Sortieren eines Feldes der Länge n, wobei $k \leq n$ ein fester Parameter ist:

- Teile das Feld in k etwa gleich große Teile.
- Sortiere die k Teilfelder rekursiv.
- Mische die k sortierten Teilfelder zu einem sortierten Feld.

Folgende drei Teilaufgaben sind zu lösen:

a) Beschreiben Sie eine Prozedur zum Mischen von k sortierten Teilfeldern.

b) Analysieren Sie diesen Algorithmus mit Ihrer Mischprozedur.

c) Vergleichen Sie die Algorithmen für verschiedene k (z.B. konst., $\log n$).

Hinweis: Sie dürfen bei der Analyse annehmen, daß n eine Potenz von k ist, also $n = k^m$.

Hausaufgabe 3 (2 + 1 + 2 Punkte)

Geben Sie für den Quicksort Median-of-three und den normalen Quicksort Algorithmus jeweils größenordungsmäßig die Anzahl benötigter Vergleiche an, falls die Eingabe ...

a) ... aufsteigend ...

b) ... absteigend ...

c) ... bis zur Mitte absteigend, dann aufsteigend ...

sortiert ist (aufeinanderfolgende Elemente unterscheiden sich höchstens um eins).

Hausaufgabe 4 (5 Punkte)

Implementieren Sie den Quickselect Algorithmus (siehe Tutoraufgabe 1) in Java.
Submittieren Sie das Programm an SOLEA unter dem Namen `QuickSelect`. SOLEA wird am Freitag den 23. Juni 2000 im Laufe des Tages freigeschaltet.

Format der Eingabe: Die erste Zeile enthält die Anzahl der Zahlenreihen. Die einzelnen Zahlenreihen beginnen mit einer Zeile, die die Anzahl der Zahlen n gefolgt von dem Rang k (es wird bei 0 begonnen zu zählen) der gesuchten Zahl enthält.

Format der Ausgabe: Geben Sie bitte für jeden Lauf des Algorithmus (sprich für jede der Zahlenreihen) das Element mit dem Rang k aus.

Testen Sie das Programm auch mit grossen Eingaben (10^6 Zahlen). Sie sollten also insbesondere Acht geben, daß die lineare average-case Laufzeit eingehalten wird. Benutzen Sie zur Beschleunigung der Einleseroutine einen `BufferedReader` bzw. `BufferedInputStream`.
Beispiel:

```
2
20 11
2
12
3
34
5
453
5
65
77
33
2
1
4
6
7
8
4
```

```
67
23
444
4 2
1
2
3
4
```

Dieses Beispiel sollte die Ausgabe

```
12
3
```

produzieren.