Prof. Dr. G.Timmer
Hochschule Osnabrück

Aufgabenblatt 1 - Quicksort

Aufgabe 1

a. Implementieren Sie den **Quicksort** aus der Vorlesung in C++. Verwenden Sie dabei die in der Vorlesung vorgestellte Lomuto-Partition. Es sollen Felder mit ganzen Zahlen sortiert werden, die aus einer Textdatei einzulesen sind. Die Eingabedaten sollen zeilenweise vorliegen. Die Ausgabe des sortierten Feldes soll ebenfalls in Textdatei geschrieben werden,
b. Generieren Sie Testdaten für das obige Programm für Feldlängen n_1=100000, n_2=1000000 und n_3=5000000 mit zufälligen Werten aus dem Intervall {0,...,n_i}.
c. Messen Sie die Zeit, die Ihr Programm für die Sortierung der Folgen der Länge n_1, n_2 und n_3 benötigt.
d. Wenden Sie nun Ihr Programm auf die zuvor sortierten Folgen an und dokumentieren Sie die Ergebnisse.

Aufgabe 2

a. Bei der Lomuto-Partition wird als Pivot-Element immer das mit dem größten Index im Feld verwendet (das auf der rechten Seite=r). Passen Sie dieses Partionsverfahren so an, dass als Pivot-Element das der Größe nach mittlere der drei Element bei den Indizes l, r und m=(l+r)/2 verwendet wird. Dieses Element kann durch Vertauschungen wieder an die Stelle r gebracht werden, damit die weiteren Schritte der Partition unverändert ablaufen können.
b. Messen Sie die Zeit, die das Programm für die Sortierung der Folgen mit den Feldlängen n_1, n_2 und n_3 benötigt und vergleichen Sie die Zeiten mit den Resultaten aus Aufgabe 1.

Hinweise:

- In der Vorlesung wurde die rekursive und iterative Variante des Algorithmus vorgestellt. Es bleibt Ihnen überlassen, welche Variante Sie einsetzten.
- Zur Generierung der Testdaten können Sie die Klasse **`Random`** (zu finden unter OSCA) verwenden. In folgendem Code-Fragment werden **`n`** Zufallszahlen im Intervall **`0...(n-1)`** generiert:

```
Random r(n);
for(int i=0; i<n; i++ ) {
    a[i] = r.give();
}
```

- Zur Zeitmessung können Sie die Funktion **`clock()`** aus **`time.h`** verwenden.
  ```
  clock_t start = clock();
  ...
  clock_t end = clock();
  time = ((end-start)*1000)/CLOCKS_PER_SEC;
  ```
  Dabei sollte natürlich nur der Aufwand für das Sortieren gemessen werden.
- Unter Umständen brechen die Programme bei großen Feldern mit Speicherfehlern ab. Dies hängt damit zusammen, dass das Betriebsystem jedem Prozess nur voreingestellte Menge von KBytes für den Stack (auf dem lokale Variablen und Felder abgelegt werden) zur Verfügung stellt. Diese Menge kann durch das Kommando **`ulimit -s <`*`Anz.KBytes`*`>`** verändert werden. Der Aufruf **`ulimit -s`** zeigt den aktuellen Wert an.

Testat: **Mo. 13.11 – Do. 16.11.**