Dr. Peter Sanders sanders@mpi-sb.mpg.de **Wintersemester 1998/99**

Übungen zur Vorlesung „Praxis des Programmierens“

http://www.mpi-sb.mpg.de/~sanders/courses/praxprog/praxprog.html

13. Übungsblatt — Abgabetermin 16.2.99

Aufgabe 1 (*Demo*)
In der letzten Vorlesung soll *jeder* Kursteilnehmer seinen Beitrag zum Projekt kurz vorstellen (2–5min). Zur Verfügung steht (hoffentlich) ein Rechner auf dem zumindest das integrierte Produkt läuft und mit einem Datenprojektor verbunden ist. Zusätzliche Erläuterungen können mit Hilfe von Folien auf einem Overheadprojektor gemacht werden. Folien und Stifte können bei Peter Sanders oder den Bremsern ausgeliehen werden. Innerhalb der Teilprojektgruppe sollten die einzelnen Vorträge koordiniert werden. Kleinere Gruppen, die eng zusammengearbeitet haben (z.B. an einem bestimmten Optimierer) können Ihren Vortrag natürlich gemeinsam halten. (Aber der Beitrag jedes einzelnen Teilnehmers sollte deutlich werden.)

Aufgabe 2 (*Merging (keine Abgabe)*)
Implementieren Sie merge-sort mit Hilfe der in der Vorlesung vorgestellten Mischroutine. Vergleichen Sie die benötigte Ausführungszeit um eine Million `ints` zu sortieren mit der Ihres LEDA-basierten Programm vom Aufgabenblatt 3.

Aufgabe 3 (*Cache-Analyse (keine Abgabe)*)
Gegeben Sei ein Cache der Größe 2^{21} Bytes und cache lines der Größe 64 Bytes. Sie wollen zwei 1024×1024 Matrizen (aus `doubles`) multiplizieren. Vergleichen Sie die Anzahl Cache misses der folgenden beiden Algorithmen aus der Vorlesung: Der einfache Algorithmus mit Spaltenpuffer. Der geblockte Algorithmus mit Blockgröße 256×256. (Sie Interference Misses vernachlässigen, d.h. so tun als hätten Sie einen fully associative Cache.) Gibt es eine noch bessere Blockgröße? Mit welchen algorithmischen Maßnahmen kann man sicherstellen, daß die Matrixmultiplikation auch bei einem direct mapped Cache eine ähnliche Leistung erzielt?