Universität Innsbruck - Institut für Informatik
Prof. Clemens Ballarin, Robert Binna, Friedrich Neurauter, Francois Scharffe and Sarah Winkler

17 June 2008

Proseminar Algorithmen und Datenstrukturen

Exercise Sheet 12

Exercise 1 (Combined Sorting)

In the last exercise sheets you implemented several different sorting algorithms. All these sorting problems had one thing in common, they used the same algorithm for the whole problem, independent of the problem size. Another strategy for sorting is to use a divide and conquer algorithm for large problems and when the problem gets small enough the algorithm switches to more suitable one. The decision what algorithm is used is made during the sorting process after a certain subproblem size was reached.

a) Extend the sorting framework from the previous exercises with the ability to measure the runtime of the different algorithms. (Hint use gettimeofday for measuring time).

b) Try different sorting algorithms for different problem sizes and measure what algorithm is suitable for which problem size. Give an explanation of your results.

c) Use the results from above to implement a combined sorting algorithm that uses a divide and conquer based algorithm for large problem sizes and a simpler, more suitable algorithm for smaller subproblems. Set the threshold for the problem size according to your results from the exercise above. Integrate your algorithm into the existing framework from the previous exercises.

d) Compare the execution time of the new combined sorting algorithm with the other “Text in standalone” algorithms.

Exercise 2 (Choose the right sorting algorithm)

Given are several scenarios where sorting algorithms are applied. Choose an appropriate sorting algorithm and explain why the algorithm was chosen:

a) Given is a list of people sorted by their year of birth. Sort them by their resting pulse rate. (Hint: the higher the age the lower the resting puls rate)

b) Given is a cryptographic algorithm that facilitates a sorting algorithm as part of its implementation. Which algorithms can prevent timing attacks?

c) Given is an alphabetic sorted list of students of the university of Innsbruck. Find an appropriate algorithm for sorting the list by the first two digits (year) of the matriculation number.

Exercise 3 (Hashtables)

a) Give the pseudo code for searching and inserting into a hash table with open addressing. Within the pseudo code $h(k, i)$ represents the hash function and m represents the size of the hash table. Think about how empty can be represented within the hashtable.

b) Construct a hash table for the following values by hand: [5, 10, 20002, 40, 2512, 3480, 97, 31]. Use open addressing with linear- and quadratic probing and double hashing. Use $m = 11$ as the size of the hashtable and $h(x) = x \; mod \; 11$ as the hashing function. Assume for quadratic probing $c_1 = 0$ and $c_2 = 1$. For double hashing assume $h2(k) = 1 + k \; mod \; 10$ as the second hash function.

Exercise 4 (Hashing and C-Compiler)

Assume you want to write a C-compiler. Every C-program consists of identifiers (function-, variable names). As the compiler needs to find identifiers fast, a hash table would be the right choice for storing such identifiers.

a) Estimate an appropriate size for the hash table depending on the size of the input.

b) Create a hash function that is able to convert the given identifiers into hash codes.

c) Find reasons why the hashing could have a bad runtime behaviour? Give an example program that will have these problems.