I/O-efficient Algorithms and Data Structures

Lars Arge

MADALGO*
Department of Computer Science
University of Aarhus
Aarhus, Denmark

large@madalgo.au.dk
www.madalgo.au.dk/∼large

Abstract

In many modern applications that deal with massive data sets, communication between internal and external memory, and not actual computation time, is the bottleneck in the computation. This is due to the huge difference in access time of fast internal memory and slower external memory such as disks. In order to amortize this time over a large amount of data, disks typically read or write large blocks of contiguous data at once. This means that it is important to design algorithms with a high degree of locality in their disk access pattern, that is, algorithms where data accessed close in time is also stored close on disk. Such algorithms take advantage of block transfers by amortizing the large access time over a large number of accesses. In the area of I/O-efficient algorithms the main goal is to develop algorithms that minimize the number of block transfers (I/Os) used to solve a given problem.

These lectures will cover I/O-efficient algorithms and data structures for a few fundamental problems in computational geometry and graph theory, with focus on the techniques used to design such algorithms. The problems include sorting and fundamental searching problems, as well as the list ranking graph problem. Most of the covered material is described in [1, 2, 3].

References

[1] A. Aggarwal and J. S. Vitter. The Input/Output complexity of sorting and related problems. *Communications of the ACM*, 31(9):1116–1127, 1988.

[2] L. Arge. External-memory geometric data structures. Lecture notes (2005) available at http://www.daimi.au.dk/∼large/ioS06/ionotes.pdf

[3] N. Zeh. I/O-efficient graph algorithms. Lecture notes (2002) from EEF Summer School on Massive Datasets available at http://users.cs.dal.ca/∼nzeh/Teaching/Summer2007/6104/Notes/zeh02.pdf

*Center for Massive Data Algorithmics — a center of the Danish National Research Foundation.