UW/CSE Programming Abstractions
Course Description
Draft of May 9, 2009

Structural place in the curriculum

- 4 credits (3 weekly lectures, 1 weekly section)
- Pre-requisites: Foundations I (for induction, logic, proofs, etc.)
- Taken by: All students (CS and CE)
- Catalog description: To be determined

Course Overview / Goals

The core of this course is fundamental "classical" data structures and algorithms including hashtables, sorting, priority queues, graphs and graph algorithms like shortest paths, etc. The course includes asymptotic complexity (e.g., big-O notation).

The course also includes an introduction to concurrency and parallelism grounded in the data structure material. Concurrent access to shared data motivates mutual exclusion. Independent subcomputations (e.g., recursive calls to mergesort) motivate parallelism and cost models that account for time-to-completion in the presence of parallelism.

The similar structure of many embarrassingly parallel tasks motivates reusable programming abstractions in the "map reduce" style in which the data traversal and the computation performed on the data are separate software components. Other approaches to processing very large data sets may also be appropriate.

More general goals of the course include (1) exposing students to non-obvious algorithms (to make the point that algorithm selection and design is an important and non-trivial part of computer science & engineering) and (2) giving students substantial programming experience in a modern high-level programming language such as Java (to continue developing their software-development maturity).

Description of Possible Homework, Etc.

The course would probably have a mix of homeworks, projects (essentially larger homeworks), and exams.

Possible Textbook

To be determined. Many publishers now offer custom books comprised of chapters drawn from multiple textbooks in their catalog. That could be a good option for this course.

Approximate Topic List

Note: While this topic list is also a plausible *order*, it is not necessarily intended to be the order. It may work better to consider concurrency or parallelism earlier or in a more interspersed way. Topics like "thread-programming basics" may be relegated to section.

Note: This list is a *first draft*, particularly toward the end.

1. Abstract data types, review of stacks and queues
2. Asymptotic analysis
3. Review of recurrence relations in context of asymptotic analysis
4. Priority queues: binary heaps, array implementations (no fancy heaps)
5. Balanced Binary search trees: e.g. AVL or red/black trees

6. B-Trees
7. Hashing (no emphasis on different forms of chaining)
8. Sorting
9. Graph representations
10. Graph search: depth- and breadth-first search
11. Greedy Graph Algorithms: Minimum spanning trees, Shortest paths
12. Thread-programming basics: thread-creation, shared-memory, multiple stacks, work queues
13. Parallel maps (for-each loops) over arrays and trees (but not linked lists)
14. Parallel reductions over arrays and trees (e.g., summing); parallel prefix
15. Speedup with infinite processors or P processors; total work vs. critical path; Amdahl's Law
16. Abstracting parallel processing via map and reduce callbacks
17. Concurrent access to shared data (e.g., a hashtable); mutual exclusion via locking
18. Coarse vs. fine-grained locking (e.g., one lock per hashtable bucket); races and deadlocks
19. Abstracting parallel searching via SQL-style queries
20. Object / relational impedance mismatch

Non-Topics and Relation to Old Curriculum

One can view this course as a modernization of CSE326: Data Structures with substantial reduction of the material on classical data structures to make room for additional topics. It still covers asymptotic complexity and fundamental data structures. Here are lecture topics from a recent offering of CSE326 that were trimmed out in this course description:

- Leftist Heaps, Skew Heaps (leaving only 1–2 priority-queue implementations)
- Binomial Queues
- Splay trees, KD trees (leaving only AVL or red-black trees and B Trees)
- Disjoint-sets (union/find) Becomes an optional topic when discussing minimum spanning trees.
- Network Flow
- Cursory mention of NP-completeness (covered more thoroughly in Foundations II)

Together these topics occupy about 7–8 lectures in that offering of 326.