

Fall 2008  
Georgia Tech, CSE 6140  
Tu/Th 8:05am - 9:25am  
Classroom: KACB Room 2447

## Computational Science & Engineering (CSE) Algorithms

<http://www.cc.gatech.edu/~bader/COURSES/GATECH/CSE6140-Fall12008/>

**Instructor:** Prof. David A. Bader, KACB 1332, 404-385-0004, bader@cc

**Office Hours:** Tuesday/Thursday 9:30am-10:30am

**Teaching Assistant:** Xin Sun, [xinsun@cc.gatech.edu](mailto:xinsun@cc.gatech.edu)

**TA Office Hours:** Friday 4PM-6PM, Klaus Room 1305

**Textbooks:**

- **Required:** *Petascale Computing: Algorithms and Applications*, Edited by David A. Bader, Chapman & Hall/CRC Computational Science Series, 2007.
- **Required:** *Parallel Processing for Scientific Computing*, Edited by Michael A. Heroux, Padma Raghavan, and Horst D. Simon, SIAM, 2006.
- **Recommended:** Cormen, Leiserson, Rivest, and Stein, *Introduction to Algorithms, Second edition*, MIT Press, 2001.

**Course Description:** This course will introduce students to designing high-performance and scalable algorithms for computational science & engineering applications. The course focuses on algorithm design, complexity analysis, experimentation, and optimization, for important science and engineering applications. Students will develop knowledge and skills concerning:

- the design and analysis of real-world algorithms employed in computational science and engineering applications, and
- performance optimization of applications using the best practices of algorithm engineering.

**Pre-requisites:** design and analysis of algorithms (CS 3510).

Students (from the Sciences, Engineering, and Computing) interested in algorithmic applications in science and engineering are encouraged to take this course.

This course can be taken for satisfying the theory breadth requirement by computer science graduate students (M.S. and non-theory Ph.D. students). This course cannot be taken by ACO students to satisfy their core requirement and theory Ph.D. students in computer science to satisfy the theory breadth requirement.

**Grading:**

- (25 %) Midterm
- (25 %) Final
- (25 %) Project
- (20 %) Homework
- ( 5 %) Class participation

## CLASS POLICIES

1. Class announcements will be sent to the Georgia Tech T-Square mailing list, see <http://t-square.gatech.edu/>.
2. Please let me know as soon as possible if you will need to re-schedule an exam, or have any special needs during the semester.
3. Each student must read and abide by the Georgia Tech Academic Honor Code, see [www.honor.gatech.edu](http://www.honor.gatech.edu).
4. Plagiarizing is defined by Webster's as "to steal and pass off (the ideas or words of another) as one's own: use (another's production) without crediting the source." If caught plagiarizing, you will be dealt with according to the GT Academic Honor Code.
5. When working on homework, you may work with other students in the class. However; you must turn in separate copies of the homework with the following written on it: your name and the names of everyone with whom you collaborated.
6. Homework is due by the end of lecture on the given due date. Late homeworks will not be accepted without a legitimate excuse and approval from the instructor.
7. No collaboration is permitted on exams. The midterm and final exams will be in-class, closed-book exams. You will be allowed to take a "cheat sheet" (double-sided 8.5 x 11 sheet of paper) into each exam.
8. Unauthorized use of any previous semester course materials, such as tests, quizzes, homework, projects, and any other coursework, is prohibited in this course. Using these materials will be considered a direct violation of academic policy and will be dealt with according to the GT Academic Honor Code.
9. For any questions involving these or any other Academic Honor Code issues, please consult me, my teaching assistants, or [www.honor.gatech.edu](http://www.honor.gatech.edu).

## Coverage of Topics

The course balances the use of theory and practice in algorithm design by presenting the student with case studies from application domains. After an introduction to computational science and engineering applications, algorithm theory, and realistic models of computation, the course consists of two modules. The first focuses on the design and analysis of real-world algorithms, with an application walk-through from algorithmic techniques to construction of the algorithm and finally application-level performance. The second module focuses on attaining high-performance implementations through algorithm engineering. These modules develop the theoretic framework and complements the study with examples from real-world problems using implementations on modern computing systems.

### Computational Science & Engineering Introduction

1. Overview of Computational Sci. and Engr. Applications; characteristics and requirements
2. Computability and Complexity
3. Ideal and Realistic Models of Computation
4. Design of Parallel Algorithms
5. Performance Analysis
6. Multi-scale, multi-discipline applications

### Real-World Algorithms: Techniques to Applications

1. Algorithms for Genomics: Divide and Conquer, cache-aware data structures, string algorithms
2. Sequence similarity searching: dynamic programming, string algorithms, local alignments, BLAST
3. Parallel Sorting by Regular Sampling: Randomized algorithms, load balancing, partitioning, merging, and sorting
4. Scientific visualization: greedy algorithms, 3d surface construction, marching cubes
5. Graph partitioning: NP-completeness, approximation algorithms, adaptive mesh refinement
6. Graph analysis: search algorithms, semantic networks, betweenness centrality

### Algorithm Engineering

1. Modeling of the Memory Hierarchy
2. Cache-friendly, cache-aware, and cache-oblivious algorithms
3. Ideal and realistic models of parallel computation
4. Parallel and distributed algorithms and applications
5. Parallel disk models
6. Problem solving frameworks
7. Tools for Performance analysis
8. Experimental methods and validation

## Fall 2008, Tentative Course Schedule

| Week | Date   | Lec | Topic  |
|------|--------|-----|--|
| 1    | 19 Aug | 1   | Intro. to Computational Science & Engineering                            |
|      | 21 Aug | 2   | Modern Computer Architecture, Models of Computation, Multicore           |
| 2    | 26 Aug | 3   | Projects   |
|      | 28 Aug | 4   | Real-world algorithm analysis  |
| 3    | 2 Sep  | 5   | Performance Analysis, High-Performance Computing                         |
|      | 4 Sep  | 6   | Realistic models of computation, Disk models, cache-oblivious algorithms |
| 4    | 9 Sep  | 7   | Practical Graph Algorithms (Guest Lecturer)                              |
|      | 11 Sep | 8   | Multi-scale, Multi-physics applications (Guest Lecturer)                 |
| 5    | 16 Sep | 9   | Review of Computability and Complexity                                   |
|      | 18 Sep | 10  | Petascale CSE Applications   |
| 6    | 23 Sep | 11  | IBM Cell processor and algorithms  |
|      | 25 Sep | 12  | Graph Partitioning (with Nodal Coordinates)                              |
| 7    | 30 Sep | 13  | Graph Partitioning (Spectral Methods)                                    |
|      | 2 Oct  | 14  | Multi-Grid Methods   |
| 8    | 7 Oct  | 15  | <b>Midterm</b>   |
|      | 9 Oct  | 16  | Parallel Sorting by Regular Sampling                                     |
| 9    | 14 Oct | -   | <b>Fall Break</b>  |
|      | 16 Oct | 17  | Large-scale Graph and Multithreaded Algorithms                           |
| 10   | 21 Oct | 18  | Real-world Connectivity Algorithms (Routing, Distance)                   |
|      | 23 Oct | 19  | Real-world Centrality Algorithms (Informatics and Interactions)          |
| 11   | 28 Oct | 20  | Biological Sequence Alignment and Searching                              |
|      | 30 Oct | 21  | Computational Biology/Phylogeny  |
| 12   | 4 Nov  | 22  | Realistic models of parallel computation                                 |
|      | 6 Nov  | 23  | Vectorized Algorithms and Techniques                                     |
| 13   | 11 Nov | 24  | Productivity Metrics and Frameworks                                      |
|      | 13 Nov | 25  | Algorithm Engineering and Performance                                    |
| 14   | 18 Nov | 26  | Project Presentations  |
|      | 20 Nov | 27  | Project Presentations  |
| 15   | 25 Nov | 28  | Experimental Methods and Validation                                      |
|      | 27 Nov | -   | <b>Thanksgiving Break</b>  |
| 16   | 2 Dec  | 29  | Problem Solving Environments for Discrete Algorithms                     |
|      | 4 Dec  | 30  | Special Topics: Open Problems in CSE Algorithms                          |
| 17   | 8 Dec  | -   | <b>Final Exam</b> (2:50pm-5:40pm)  |

### Additional Readings:

1. Guy Blelloch, Algorithms in the Real World, Lecture Notes
2. Alok Aggarwal and Jeffrey Scott Vitter, The Input/Output Complexity of Sorting and Related Problems, *Communications of the ACM*, 31:1116-1127, 1988.
3. Sandeep Sen, Siddhartha Chatterjee, Neeraj Dumir, Towards a theory of cache-efficient algorithms, *Journal of the ACM*, 49(6):828-858, 2002.
4. A. LaMarca and R.E. Ladner, The Influence of Caches on the Performance of Heaps, *Journal of Experimental Algorithmics*, Vol 1, 1996.
5. A. LaMarca and R.E. Ladner, The Influence of Caches on the Performance of Sorting, *Journal of Algorithms*, 31:66-104, 1999.
6. R.E. Ladner, J.D. Fix, and A. LaMarca, Cache Performance Analysis of Traversals and Random Accesses, Tenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 1999.
7. John W. Romein, Jaap Heringa, Henri E. Bal, A Million-Fold Speed Improvement in Genomic Repeats Detection, *Supercomputing 2003*.
8. Joon-Sang Park, Michael Penner, Viktor K Prasanna, Optimizing Graph Algorithms for Improved Cache Performance, *International Parallel and Distributed Processing Symposium*, Fort Lauderdale, FL, April 2002.
9. Markus Kowarschik, Ulrich Rude, Christian Weiss, and Wolfgang Karl, Cache-Aware Multi-grid Methods for Solving Poisson's Equation in Two Dimensions, *Computing*, 64:381-399, 2000.
10. M. Frigo, C. E. Leiserson, H. Prokop, and S. Ramachandran, Cache-Oblivious Algorithms, *IEEE Symposium on Foundations of Computer Science*, 1999.
11. Stephen Alstrup, Michael A. Bender, Erik D. Demaine, Martin Farach-Colton, J. Ian Munro, Theis Rauhe, Mikkel Thorup, Efficient Tree Layout in a Multilevel Memory Hierarchy, Extended version of *ESA 2002* paper, November 2002.
12. Lars Arge, Michael A. Bender, Erik D. Demaine, Bryan Holland-Minkley, J. Ian Munro, Cache-Oblivious Priority Queue and Graph Algorithm Applications, *34th ACM Symposium on Theory of Computing (STOC)*, 2002.
13. L. G. Valiant, A Bridging Model for Parallel Computation, *Communication of the ACM*, 33(8):103-111, 1990.
14. D. E. Culler, R. M. Karp, D. A. Patterson, A. Sahay, K. E. Schauer, E. Santos, R. Subramonian, and T. von Eicken, LogP: Towards a Realistic Model of Parallel Computation, *4th ACM Symp. Principles and Practice of Parallel Programming (PPoPP)*, pp 1-12, May 1993.
15. V. Ramachandran, A General-Purpose Shared-Memory Model for Parallel Computation, in M. T. Heath and A. Ranade and R. S. Schreiber (eds.), *Algorithms for Parallel Processing v 105*, pp 1-18, Springer-Verlag, New York, 1999.

16. M. Snir and D.A. Bader, A Framework for Measuring Supercomputer Productivity, The International Journal of High Performance Computing Applications, 2004.
17. D.A. Bader, B.M.E. Moret, and P. Sanders, “High-Performance Algorithm Engineering for Parallel Computation,” Lecture Notes of Computer Science, 2002.
18. William E. Lorensen, Harvey E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. Computer Graphics, Volume 21, Number 4, July 1987.
19. M. Erez, J. H. Ahn, A. Garg, W.J. Dally, and E. Darve, Analysis and Performance Results of a Molecular Modeling Application on Merrimac, SC’04, Pittsburgh, PA, November 2004.
20. D.S. Johnson, A Theoretician’s Guide to the Experimental Analysis of Algorithms, in Proceedings of the 5th and 6th DIMACS Implementation Challenges, M. Goldwasser, D. S. Johnson, and C. C. McGeoch, Editors, American Mathematical Society, Providence, 2002.