CMSC 451: Design and Analysis of Computer Algorithms

Fall 2003

http://www.cs.umd.edu/~mount/451/

Instructor: Dave Mount. Office: AVW 3373. Email: mount@cs.umd.edu. Office phone: (301) 405–2704. Office hours: Mon 2:30-3:30, Wed 3:30-4:30. I am also available immediately after class for questions. If the question is short (a minute or so) drop by my office any time. Please send me email if you cannot make these times. (Don't be shy about doing this. I always set aside at least one hour each week for "unscheduled" office hours.)

Class Time: Tue, Thur 3:30-4:45, CSI 3117.

Teaching Assistant: Pooja Nath. Office: AVW 1112. Email: pooja@cs.umd.edu. Office hours: Tue,Wed 9:00-10:00. If you cannot make these times, please feel free to contact Pooja to set up another time.

Course Overview: This course presents the fundamental techniques for designing efficient computer algorithms, proving their correctness, and analyzing their running times. After a brief review of material from 351 (asymptotics, recurrences, sorting), we will discuss efficient algorithms for basic graph problems (minimum spanning trees, shortest paths, connectivity problem, network flows), solving optimization problems through greedy algorithms and dynamic programming, computational geometry, proofs of intractability and NP-completeness, and approximation algorithms.

Text: (Required) T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms* (2nd Edition), McGraw Hill, 2002. (This is an excellent reference book on algorithm design. I strongly recommend purchasing it, and keeping it for future reference.)

Prerequisites: CMSC 351. Each student is expected to have basic programming skills (programming with loops, pointers, structures, recursion), discrete mathematics (proof by induction, sets, permutations, and combinations, probability), understanding of basic data structures (lists, stacks, queues, trees, graphs, and heaps), knowledge of sorting algorithms (MergeSort, QuickSort, HeapSort) and basic graph algorithms (DFS and BFS), basic calculus (manipulation of logarithms, differentiation, integration). If there is any material that seems unfamiliar, please see me or the teaching assistant as soon as possible to head off any problems.

Course Work: Course work will consist of (around 6) homework assignments (about one every week and a half) and 2 exams (a midterm and a comprehensive final). Tentative weights: Homeworks 25%, midterm 30%, final exam 45%. (Note that these weights are subject to change.) The midterm is tentatively scheduled for Tue, Oct 28 and the final exam will be Sat, Dec 20, 10:30am-12:30pm.

Homeworks are to be turned by the start of class on the due date. No late homeworks will be accepted, but the lowest homework grade will be dropped. (In other words, turn in what you have by the start of class. If your schedule is too busy, just drop the homework. But, try your best to save the dropped homework for the latter half of the semester, when you will really need it.) In exceptional circumstances (illness, university business, religious observances) extensions may be granted. However, all extensions must be approved by me BEFORE the due date.

As a courtesy to the grader, homeworks should be written neatly. Poorly written work will not be graded. When writing algorithms be sure not only that your solution is correct, but also that it is easy for the grader to understand why your solution is correct. Part of your grade will be based not only on correctness, but also on the clarity, simplicity, and elegance of your solution.

Some homeworks and projects will have a special challenge problem. Points from the challenge problems are *extra credit*. This means that I do not consider these points until *after* the final course cutoffs have been set. Each semester extra credit points usually account for at least few students getting one higher letter grade (e.g. from a B+ to A-).

Group Homeworks: Some people learn better in a group setting and others learn better individually. This semester we will experiment with a group homeworks. For each homework assignment, you are allowed to work either individually or in groups of two or three people. You pick your own group members, and may use different groups for different homeworks. Each group shall hand in one homework with the names of all the people of the group, and all members of the group will share the same grade. Grading standards will be the same, irrespective of whether the assignment is done individually or in a group, but I will expect group homeworks to be neatly written and well organized (because there is one document to be prepared for the entire group).

It may seem at first that groups will have an obvious advantage over people working individually. However, this is not true for a couple of reasons. First, when working in a group it is important the final document reflect a common vision of the group, resolving any differences among the group members. Second, the exams (which are worth much more than the homeworks) require on a deep understanding of the methods used in solving homework problems. Group members that only have a superficial understanding of the material will pay a heavy price on the exams. In-depth knowledge only comes by struggling (often for hours) with different approaches and solution strategies.

Academic Dishonesty: You may *discuss* homework problems and general solution strategies with classmates (whether inside or outside your group), but when it comes to formulating and writing solutions you must work alone or only with your fellow group members. If you make use of other sources in coming up with your answers you must cite these sources clearly. (This includes papers or books in the literature, friends or classmates, and information downloaded from the web.) Instances of academic dishonesty will be dealt with harshly, and usually result in a hearing in front of a student honor council, and a grade of XF.

Syllabus: The topics and order listed below are tentative and subject to change.

Review of algorithm analysis and sorting: (Chapts 1–9) Review of asymptotics, summations, recurrences, sorting algorithms, selection, and lower bounds for sorting. (1 week)

Dynamic Programming and Greedy Algorithms: (Chapts 15–16) Longest common subsequence, chain multiplication, minimum weight triangulation, Huffman trees. (2 weeks)

Basic Graph algorithms: (Chapts 22–23) Review of basic graph traversals (DFS and BFS), directed acyclic graphs, topological sort and strong components, minimum spanning trees. (2 weeks)

Shortest Paths: (Chapt 24) Dijkstra's algorithm and Bellman-Ford algorithm. (1 week)

Network Flow: (Chapt 26) Ford-Fulkerson and Edmonds-Karp algorithms, and applications to maximum matching. (1 week)

Computational Geometry: (Chapt 33) Closest pair, convex hulls. (1 week)

NP-completeness: (Chapts 34) Basic terminology, polynomial reductions, examples of NP-complete problems (SAT, independent set, vertex cover, clique, Hamiltonian path, TSP), approximation algorithms. (2.5 weeks)

Approximation Algorithms: (Chapt 35) Vertex cover, TSP k-center approximations. PTASs and the knapsack approximation. (1.5 weeks)