Course Overview: This course presents the fundamental techniques for designing efficient computer algorithms, proving their correctness, and analyzing their complexity. General topics include graph algorithms, basic design paradigms (such as divide-and-conquer, dynamic programming and greedy algorithms), network flow and NP-completeness.


Prerequisites: CMSC 351. Each student is expected to know the basic concepts of programming (e.g. loops, pointers, recursion), discrete mathematics (proof by induction, sets, permutations, probability), simple data structures (lists, stacks, queues, trees, heaps), basic calculus (logarithms, differentiation, integration). If there is any material that seems unfamiliar, please see the instructor or the teaching assistant as soon as possible to head off any problems.

Course Work: Course work will consist of 6 homework assignments and two exams (a midterm and a comprehensive final). The midterm date will be posted on the class web page. The final will be on Friday, August 20, the last day of class.

You have to turn your homework in at the beginning of class on the due date. Since homework solutions will be discussed in class the day the homework is due no late homework will be accepted. If you cannot come to class the day the homework is due you must turn in your homework before the due date.

All homeworks are to be done independently, with no help from the web or other sources. If you have questions, please talk to the TA or the instructor. Assignments are to be written up neatly; poorly written homeworks will not be graded.
Grading: Final grades will be based on homework assignments, the midterm exam, and the comprehensive final exam. Tentative weights for these will be 25% for the homeworks, 35% for the midterm and 40% for the final. Note that these weights are subject to change.

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

1. Basic Graph algorithms: basic graph traversals (DFS and BFS), strongly connected components, topological sort (4 lectures).
2. Greedy Algorithms: minimum spanning tree, shortest path, scheduling, matroids (5 lectures)
3. Divide and conquer: convex hull, median finding (2 lectures)
4. Dynamic programming: optimal binary search trees, shortest paths, matrix chain multiplication (4 lectures)
5. Network flow: Ford-Fulkerson and Edmon-Karp algorithms, maximum matching (5 lectures)
6. NP-completeness: basic terminology, polynomial reductions, NP-complete problems (SAT, independent set, vertex cover, clique, Hamiltonian path, TSP.) (4 lectures)
7. Approximation algorithms: matching, vertex cover, k-center. (2 lectures)